Издательство «Знание»

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Новое в жизни, науке, технике

Подписная научно-популярная серия

Издается ежемесячно с 1988 г.

Универсальный язык – возможно ли это?

1988

8

Новое
в жизни,
науке,
технике

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Подписная
научно-
популярная
серия

8/1988

Издается
ежемесячно
с 1988 г.

УНИВЕРСАЛЬНЫЙ ЯЗЫК — ВОЗМОЖНО ЛИ ЭТО?

В НОМЕРЕ

„КРУГЛЫЙ СТОЛ“ СЕРИИ

3 **ПРОБЛЕМЫ РАЗРАБОТКИ УНИВЕРСАЛЬНОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ**

ДЕДКОВ А. Ф., ЩЕРС А. Л.
ЖУРАВЛЕВ А. П.
ВАСИЛЬЕВ П. П., ФРОЛОВ С. А., МАЛЫШЕВ С. И.
РОДИОНОВ А. Б.
ЛАВРОВ С. С.

РУБРИКИ

12 Языки программирования

32 Отвечаем на письмо читателя

35 Базовые модели школьной информатики

42 Терминал. Компьютерный клуб школьников

Издательство
„Знание“
Москва
1988

ББК 32.973—01
У 59

Авторы выпуска

ДЕДКОВ Анатолий Федорович — кандидат технических наук, заведующий сектором ВНИИ проблем вычислительной техники и информатики.

ЩЕРС Артур Львович — доктор технических наук, профессор, заведующий отделением ВНИИ проблем вычислительной техники и информатики.

ЖУРАВЛЕВ Александр Павлович — доктор филологических наук, профессор Московского института народного хозяйства им. Г. В. Плеханова.

ВАСИЛЬЕВ Петр Павлович — кандидат экономических наук, старший научный сотрудник ВМЦ „Интеграл“.

ФРОЛОВ Степан Алексеевич — кандидат технических наук, начальник отдела ВМЦ „Интеграл“.

МАЛЫШЕВ Сергей Иванович — кандидат технических наук, начальник сектора ВМЦ „Интеграл“.

РОДИОНОВ Андрей Борисович — работает в Институте проблем информатики АН СССР.

ЛАВРОВ Святослав Сергеевич — член-корреспондент АН СССР.

МАЛЫХИНА Мария Петровна — кандидат технических наук, доцент.

ЧАСТИКОВ Аркадий Петрович — кандидат технических наук, доцент.

ЛОГВИНОВ Игорь Иосифович — кандидат педагогических наук, директор Главного информационно-вычислительного центра Министерства просвещения СССР.

РАТИНСКИЙ Марк Гершевич — заведующий учебно-методическим кабинетом информатики Московского областного института усовершенствования учителей, автор 12 печатных работ.

ШВЕДОВСКИЙ Вячеслав Анатольевич — кандидат физико-математических наук, работает в Институте социологических исследований АН СССР.

МАШБИЦ Ефим Израильевич — кандидат психологических наук, заведующий лабораторией Института психологии Министерства просвещения УССР.

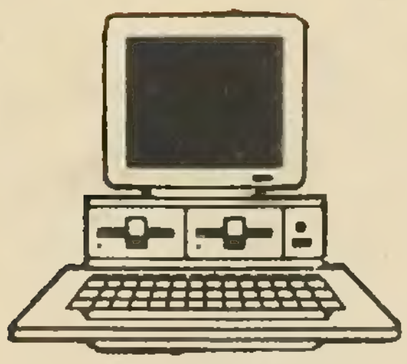
МЕТАЛИДИ Александр Сократович — инженер-программист, работает в Симферопольском Государственном университете.

ПЕРЕХОД Игорь Александрович — кандидат физико-математических наук, доцент Симферопольского Государственного университета.

КАСАТКИН Валентин Николаевич — кандидат педагогических наук, доцент Симферопольского Государственного университета.

РЕДАКТОР Б. М. ВАСИЛЬЕВ

© Издательство „Знание“, 1988 г.



Интересен сам по себе вопрос, который вынесен на обсуждение. Почему вдруг он возник и не является ли он искусственным? Его задают себе и специалисты, и руководители. Последние чаще попадают в состояние неопределенности, когда нужно сделать выбор среди множества языков. Выбор же этот, как правило, труден не только для руководителей, управленцев, но и для специалистов. Только универсальный язык для пользователя поможет уйти от существующего множества вариантов.
Следовательно, круг людей, задающих себе этот вопрос, достаточно широк, а сам вопрос вполне уместен.

ПРОБЛЕМЫ РАЗРАБОТКИ УНИВЕРСАЛЬНОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ

А. Ф. ДЕДКОВ,
А. Л. ЩЕРС. В первое десятилетие развития вычислительной техники, от которого нас отделяет треть века, программы писались на так называемых машинных языках (или в машинных кодах). Такие программы непосредственно выполняются компьютерами. Каждый тип компьютера имел уникальный машинный язык, и только в 60-е годы возникли семейства ЭВМ, обладающие единым машинным языком. Машинные языки были и остаются по сей день крайне сложными и неудобными для человека. И только после того как был накоплен первый опыт применения машин для решения различных классов задач, было осознано, что компьютер — это не просто большой и быстрый автоматический калькулятор, но универсальный инструмент, который может выполнять любую формализованную работу по переработке информации. Именно такой задачей является перевод программы с одного формального языка на другой. Возникли первые языки программирования, которые сейчас принято называть языками высокого уровня, удобные для записи тех или иных классов алгоритмов. Для решения научных и технических задач — ФОРТРАН и АЛГОЛ-60, для экономических — КОБОЛ и др. Создаются и первые программы-трансляторы, целью разработки которых является автоматизация перевода программы, записанной на языке программирования, в эквивалентную программу на машинном языке. Но поскольку транслятор (или как сейчас принято говорить — компилятор) некоторого языка программирования может быть создан для различных типов ЭВМ, то языки высокого уровня обеспечили потенциальную возможность преодоления языкового барьера между различными типами машин.

И вот родилась идея. Если создать язык программирования, удобный для человека и одинаково хорошо пригодный для решения всех видов задач и при этом разработать компиляторы языка для всех типов компьютеров (или, как говорят, реализовать язык), задача создания удобного универсального инструмента для программирования на ЭВМ будет решена отныне и навеки.

Идея создания универсального языка программирования имеет ряд притягательных аспектов: после изучения одного-единственного языка программист-новичок становится способным писать программы для любого типа ЭВМ, читать все ранее созданные программы и «разговаривать» с коллегами на одном языке. Каждый новый компьютер после реализации на нем универсального языка сразу становится способным выполнять все ранее написанные на этом языке программы, исчезает проблема переноса программ с одного типа ЭВМ на другой.

Казалось бы, преимущества единого универсального языка программирования бесспорны, но тем не менее можно утверждать, что такого языка до сих пор не существует. Почему? Почему затрачиваются сотни миллионов на разработку и реализацию все новых и новых языков программирования (а за всю короткую историю вычислительной техники их создано несколько тысяч), на переобучение программистов, на перевод программ с одного языка на другой? Почему?

Первый и, наверное, самый простой ответ заключается в том, что универсальный язык программирования не может быть создан потому, что невозможно заранее предвидеть все новые области применения ЭВМ. Однако этот ответ не удовлетворяет нас, так как можно создать язык, который содержал бы в себе средства собственного саморазвития (расширения), приспособления языка к новым классам решаемых задач. Попытки создания таких расширяемых языков делались (наиболее известный из них — СИМУЛА-67), однако они не дали желаемых результатов.

Предположим, однако, что такой идеальный саморазвивающийся язык создан. К чему бы это привело? Включение в него все новых и новых групп языковых конструкций, ориентированных на решение все новых и новых классов задач, привело бы к тому, что он стал бы настолько сложным и громоздким, что одному человеку было бы не под силу изучить его полностью. Да программисту, работающему в той или иной области, и не нужно было бы изучать его целиком, ему достаточно было бы только той части языка, которая непосредственно нужна ему для практической работы. То есть язык, по сути дела, распался бы на массу диалектов, объединенных только единым синтаксисом. Причем программист, работающий в одной области, оказался бы почти не способным понять программу, написанную на другом диалекте. Подобное явление мы наблюдаем и в естественных языках. Из беседы двух специалистов с использованием научного жаргона слушателю-неспециалисту может быть понятно только одно: что беседа идет на русском языке, но не более того. Итак, наш гипотетический универсальный язык распался бы под собственной тяжестью на ряд почти не пересекающихся диалектов-жаргонов, и мы имели бы ситуацию в программировании, не слишком отличающуюся от той, которую имеем в настоящее время.

Сейчас мы наблюдаем непрерывный процесс эволюции языков программирования. Как и любые другие объекты во Вселенной, они рождаются, живут и умирают. Но от неживых объектов их отличает то, что языки могут давать потомство. В какой среде происходит эволюция таких формальных систем, какими являются языки программирования? Она происходит в мозгу людей, разрабатывающих и применяющих их. Причем язык программирования стал не только продуктом деятельности мозга, но и фактором, изменяющим эту деятельность. Язык программирования для программиста-профессионала — это не просто набор правил записи алгоритмов решения задач, это фактор, определяющий стиль его мышления.

Конечно, не каждый язык диктует свой уникальный стиль. Существуют семейства родственных языков, отличающихся друг от друга в синтаксических деталях и мелочах и сходных в навязываемом ими стиле программирования. Иногда и от опытных программистов приходится слышать, что не язык определяет стиль их мышления, что они могут с легкостью перейти с одного языка на другой, не изменяя свой стиль. Скорее всего это объясняется тем, что они имели дело только с одним семейством родственных языков, так как большинство программистов работает на так называемых процедурных языках. Кроме того, эволюцию языков программирования более правильно было бы рассматривать как эволюцию абстрактных идей построения языков, а рождение конкретных языков — как объединение той или иной совокупности идей в том или ином сочетании и в том или ином синтаксическом обрамлении.

Рассмотрим основные движущие

силы развития языков программирования. Существуют три основные предпосылки создания нового языка. Во-первых, стремление создать универсальный язык. Примеры: АЛГОЛ-60, АЛГОЛ-68, СИМУЛА-67, ПЛ/1. Во-вторых, желание создать специализированный язык, который для данной узкой области обеспечивал бы значительное увеличение эффективности по сравнению с уже имеющимися языками. Известно, что применение специализированных языков может сократить затраты на разработку программ для некоторой более или менее узкой области применения в десятки раз. Пример — так называемые языки имитационного моделирования сложных систем. В-третьих, необходимость расширить уже существующий язык новыми идеями, возникшими уже после его рождения. В последнем случае, правда, часто возникает не новый язык, а диалект существующего. Однако иногда влияние новых идей бывает столь сильным, требует столь радикальных изменений в языке, что диалект обретает самостоятельную жизнь и продолжает жить даже тогда, когда его предок уже завершил свой жизненный цикл.

Как рождается язык? Обычно язык создает небольшая группа специалистов, иногда даже международная рабочая группа (как это было с языком АЛГОЛ-60, АЛГОЛ-68), иногда автором языка является один человек. Так, например, автором двух всемирно известных языков программирования ПАСКАЛЬ и МОДУЛА-2 стал один человек — профессор Никлаус Вирт. Как правило, новый язык создается не на пустом месте, а в пространстве идей, развитых ранее в одном или нескольких существующих языках с добавлением нескольких оригинальных идей — «изюминок». Например, язык ПЛ/1 был порожден смешением средств трех известных языков — ФОРТРАНа, АЛГОЛа-60 и КОБОЛа с добавлением ряда новых средств и погружением их в единую синтаксическую оболочку. Бывает, однако, что новый язык почти ничего не наследует от других языков. Примеры: ЛИСП, АПЛ, ПРОЛОГ. Такому языку трудно выжить, особенно если ему не удалось сразу захватить пустующую «экологическую нишу». Но если он выжил, то может дать начало новому направлению в программировании. Комплекс заложенных в нем идей может стать свежей струей, проникающей в другие языки. Кроме того, сам он может быть подвержен влиянию других языков. При этом возникают языки, которые, если придерживаться биологической терминологии, можно назвать межвидовыми гибридами.

Для того чтобы можно было проверить достоинства нового языка на практике, необходимо выполнить его экспериментальную реализацию. Часто это делает сам автор (или авторы) языка, однако не всегда. Иногда реализация языка отдается другой группе специалистов, хотя Н. Вирт и утверждал, что такая практика неверна, так как языковые средства должны проектироваться одновременно с их реализацией. Только тогда можно достичь простоты реализации языка — важнейшего свойства.

Получит ли новый язык хотя бы ограниченное распространение или окажется мертворожденным? Это зависит от ряда факторов, в том числе и чисто психологических или даже социальных. На выбор языка для реализации того или иного программного проекта влияют не только достоинства и недостатки данного языка, но и наличие на данной машине компилятора этого языка, требования (традиции), принятые в данной организации, необходимость обеспечения переносимости программ, известность автора языка, солидность организации, разработавшей или поддерживающей язык, и даже соображения моды и престижности.

К сожалению, задача объективной оценки языка программирования затрудняется тем, что его достоинства и недостатки проявляются только во взаимодействии с человеком-программистом, пишущим на этом языке. Легко сравнить эффективность машинного кода, создаваемого компиляторами различных языков, но очень сложно — легкость изучения языка, простоту его использования, надежность создавае-

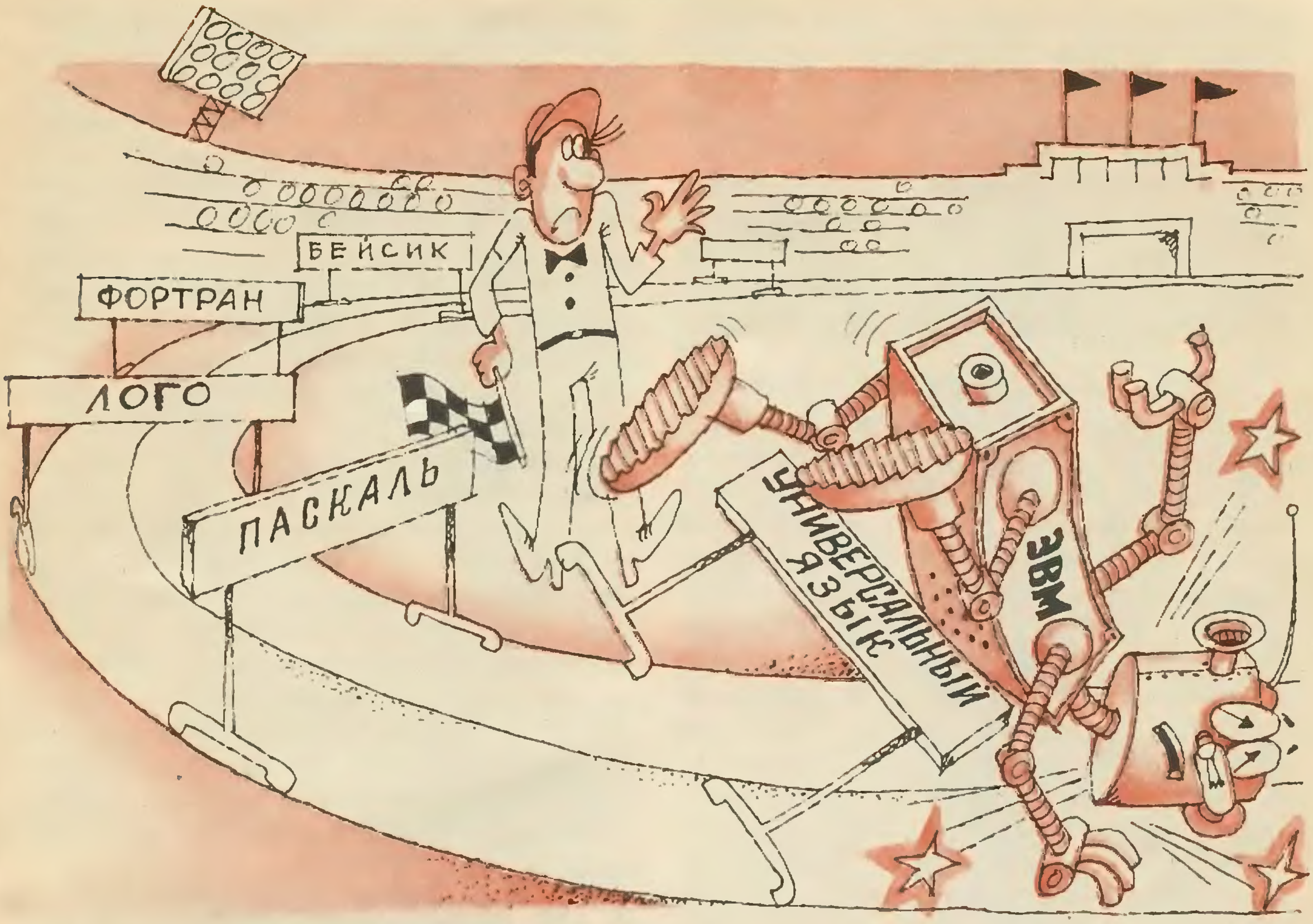
мых на нем программ. А ведь именно эти факторы становятся все более важными, так как быстрое развитие аппаратных средств вычислительной техники отодвигает вопросы эффективности на второй план. Что с того, что программа решения некоторой задачи, написанная на языке А, будет выполняться втрое быстрее, чем та же программа, написанная на языке Б (т. е. одну секунду вместо трех, если на ее написание и отладку ушло три месяца вместо одного). Экспериментальных исследований по оценке субъективных свойств языков программирования проводится крайне мало, и затрудняются они тем, что на результаты экспериментов могут сильно влиять выбор испытуемых, их квалификация, личностные особенности и даже неосознанное стремление к тем или иным результатам эксперимента. Вот почему на оценку свойств нового языка программистской общественностью так сильно влияют непроверенные слухи, общепринятые мнения и особенно оценки лиц, которые считаются экспертами в данной области.

Иногда новый язык внедряется в административно-приказном порядке, как это было, например, с известным языком АДА, который принят министерством обороны США в качестве единственного языка для всех его разработок. Но и это не всегда помогает. Например, язык ПЛ / 1, разработанный фирмой IBM для машин семейства IBM / 360 в качестве единого и универсального языка, не смог вытеснить другие языки, несмотря на поддержку такой крупной фирмы, какой является IBM.

Как уже было сказано, число разработанных языков исчисляется тысячами, а количество широко применяемых — в лучшем случае десятками. Отсюда следует, что большинство созданных языков получают лишь ограниченное распространение, т. е. применяются в одной или нескольких организациях в течение ряда лет. Однако и в этом случае нельзя считать, что разработка была выполнена напрасно. Заложенные в язык новые идеи могут впоследствии возродиться в других разработках, особенно если они были опубликованы. Часто язык, получивший достаточно широкое применение, вытесняется своими потомками, вобравшими в себя его лучшие идеи и развившими их. Так, язык АЛГОЛ-60 был вытеснен более современными языками. Однако этих языков не было бы, не будь АЛГОЛа-60, так как многие его идеи стали классикой и в том или ином виде присутствуют в вытеснивших его языках.

Если язык получил, несмотря ни на что, широкое распространение, это не означает еще, что он останется неизменным до конца срока своей жизни. Часто выпускаются новые версии языка, сохраняющие преемственность по отношению к старой версии, расширяющие ее возможности и приспосабливающие ее к новым требованиям. Так, например, язык ФОРТРАН, разработанный в середине 50-х годов, живет и сейчас, пройдя ряд модификаций. Иногда жизнь языка продлевается надолго из-за того, что он удачно вписался в некоторую экологическую нишу, заняв ее первым. Вытеснить его из этой ниши часто очень сложно, хотя преимущества других языков и очевидны. Причина этого — определенная инертность программистов и давление груза уже написанных на этом языке программ.

Рассмотрим еще один источник новых идей в области языков программирования и еще одну причину изменчивости языков. Это изменения, вносимые в язык во время его реализации. Очень часто во время реализации некоторого языка на данной машине в него вносятся небольшие изменения. Причины этого разные. Иногда это искреннее стремление реализаторов улучшить язык, исправить очевидные недостатки, обнаруженные после его первоначальной разработки, или желание приспособить язык к особенностям данного типа ЭВМ. Иногда это невозможность реализовать язык в полном объеме на данном типе ЭВМ. Тогда в язык вводятся некоторые ограничения. Иногда промышленные реализации языка одновременно ведутся различными конкурирующими фирмами. В этом случае в язык часто вносятся некоторые расширения, призванные привлечь



потенциальных покупателей и сделать данную реализацию более конкурентоспособной. Во всех этих случаях, однако, результат один: разные реализации одного и того же языка даже на одном типе ЭВМ, не говоря уж о различных типах, немного отличаются одна от другой. Различия часто незначительны, но они приводят к тому, что при переносе программы на другую машину или при переходе на другой компилятор языка программистам приходится тщательно вылавливать все отличия и исправлять программу. Хорошо, если эти различия обнаруживаются компилятором. Когда же изменены правила выполнения той или иной языковой конструкции, то случается, что программа по непонятным для программиста причинам ведет себя не так, как раньше. Поиск причин такого поведения может вылиться в длительное расследование.

Однако модификация языка не всегда приносит только одни неприятности. Часто небольшие изменения действительно значительно увеличивают возможности языка и в дальнейшем, проникая и в другие реализации, становятся фактическим стандартом.

Для того чтобы пользователь мог хоть как-нибудь ориентироваться в большом числе чуть-чуть отличающихся версий языка, введено понятие стандарта языка. Разумеется, далеко не каждый язык заслуживает чести быть стандартизованным. Обычно стандартизуются только небольшие, широко употребляемые в данной стране или в мире языки. Стандарты на язык бывают национальными и международными. Стандарт четко оговаривает синтаксис и семантику некоторого базового ядра языка. Это ядро обычно у́же, чем версии языка, реализуемые разными конкретными компиляторами. Однако в стандарт вводится требование, чтобы каждая реализация языка воспринимала средства стандартизованного ядра одинаково. Наличие стандарта дает два существенных преимущества. Во-первых, появляется основа для сравнения различных версий языка. Строго определив отличия каждой версии от стандартной (а это сделать

довольно просто), можно, не прибегая к экспериментам на машине, выяснить, какие места программы подлежат переделке при переходе на другую реализацию языка. И во-вторых, у программистов появляется реальная возможность создавать действительно переносимые программы. Для этого достаточно только не использовать никаких языковых средств, не определенных стандартом.

Какой же вывод можно сделать из всего сказанного? Появится ли, наконец, единый универсальный язык или и в будущем понадобятся самые разнообразные языки? Дать окончательный ответ на этот вопрос в настоящее время не представляется возможным. Однако хотелось бы отметить следующее. Процесс эволюции языков программирования является следствием процесса развития крайне сложной человеко-машинной системы, включающей в себя, с одной стороны, миллионы людей (пользователей, программистов) — носителей некоторого постоянно меняющегося множества идей и, с другой — миллионы компьютеров, представляющих собой материальную среду, где воплощаются эти идеи. Как любая сложная динамическая система, она способна к саморазвитию только при наличии некоторого минимального уровня разнообразия, поэтому появление сотен новых языков является необходимым условием ее интенсивного развития, а не причудой программистов и не следствием попыток «удовлетворения собственного любопытства за государственный счет». Подобно тому как в живой природе каждый вид, пусть самый незначительный, вносит свой вклад в общий генофонд биоценоза, так и почти каждый язык или некоторая языковая идея, пусть даже и не выраженная в виде полного языка, является частью общего «генофонда» программирования.

Не следует, однако, делать вывод, что стремление создать универсальные языки противостоит прогрессу этой системы и может остановить ее развитие. Напротив, эти попытки чрезвычайно плодотворны, так как при создании языка, претендующего на универсальность, переосмысливаются и более строго формулируются многие удачные идеи, разработанные по различным языкам. Каждый «универсальный» язык подводит определенный итог некоторому периоду развития этой системы и образует концептуальную основу для следующего витка ее развития. Будет ли в конце концов создан действительно универсальный язык, мы не знаем. Как всякий сложный процесс, процесс эволюции языков программирования идет по своим объективным законам, которые в настоящее время неизвестны. Для постижения этих законов требуется создать новое направление в науке, назовем его, например, «компьютерное языковедение». Оно должно охватить не только чисто языковые аспекты процесса генезиса языков программирования и технические стороны (вопросы реализации языков и влияние их на архитектуру ЭВМ, и наоборот), но и аспекты экономические, психологические и даже социальные.

А. П. ЖУРАВЛЕВ. Казалось бы, все ясно: разнообразие человеческих языков возникло потому, что они развивались стихийно. Никто этот процесс не планировал, не направлял, каждый язык развивался в своей «экологической нише», отдельно от других, и потому каждый пошел своим путем. Прекрасная иллюстрация — кавказские языки: что ни долина, то свой язык. Оно и понятно. Ведь как было в древности: долина со всех сторон окружена горными хребтами, большую часть года непроходимыми, да еще постоянная вражда племен из-за земли, из-за пастбищ... Вот и замкнулся каждый долинный язык в себе. А в результате сейчас их там десятки, и все разные.

С языком компьютера опять, казалось бы, все ясно: язык для него создал человек. Делает он это, вроде бы, планово, а следовательно, никак не должен повторять «вавилонской» ошибки, не должен создавать языковых барьеров еще и между компьютерами. А тем не менее не прошло и полувека, как появилось множество компьютерных языков и их версий и продолжают кон-

струироваться все новые.

Но может быть, это первая стадия, на которой идет конкуренция различных решений, идет отбор наилучшего из компьютерных языков, который в конце концов выявится, победит в соревновании и останется единственным, всеобщим?

Тогда возникает вопрос: что значит наилучший из компьютерных языков, каков критерий его оценки? А критерий один — компьютерный язык тем лучше, чем он ближе к естественному языку. Развитие машинных языков как раз и идет по линии их приближения к человеческому.

Вот мы и вернулись в «Вавилон». К человеческому-то к человеческому, да только к какому?! Ведь языков сейчас на земле тысячи. К какому же из них приближать компьютерный?

Здесь уместно несколько приостановиться и детальнее разобраться в машинных языках. Сначала рассмотрим проблему компьютерного языка с позиций интересов программиста.

В любом случае в настоящее время компьютер работает не менее чем на четырех языках. Во-первых, это «язык» реальных машинных кодов, «алфавит» которого состоит всего из двух букв: наличие сигнала — отсутствие сигнала.

Во-вторых, компьютеру нужна операционная система. Это как бы внутренний язык компьютера, даже как бы часть самой машины. В-третьих, существует язык-транслятор, который нужен для обеспечения работы программиста. Если бы программист составлял программу в машинных кодах, как и было на заре компьютеризации, то это потребовало бы от него колоссальных затрат труда и времени. Правда, такое программирование наиболее виртуозно, поскольку программист разговаривает с машиной непосредственно на ее языке. Иногда высококлассные программисты прибегают к программированию в машинных кодах, если не хватает возможностей конкретного языка программирования.

Но все же в основном сейчас все компьютеры оснащаются операционными системами и трансляторами, на которых «думает» компьютер. Так вот, сейчас компьютеры, в том числе и наши, отечественные, «думают» в основном по-английски. Правда, операционные системы и трансляторы, построенные на основе английского, только с большой натяжкой можно назвать английским языком. Это даже не уродливый «пиджин-инглиш», на котором можно объясниться почти в любом морском порту мира на уровне «твоя — моя не понимай», это гораздо более примитивные языки. И все же в их основе — английский.

Наконец, в-четвертых, существуют конкретные языки программирования, на которых программист пишет программу. Наиболее известные из них — БЕЙСИК, ПАСКАЛЬ, ФОРТРАН — все на основе английского. Так что практически программисты мира, в том числе и русские, беседуют с компьютером по-английски.

Для англоязычных программистов это, конечно, преимущество перед другими, ведь всем прочим приходится еще в какой-то мере учить английский. Правда, труд не бог весть какой — алфавит, да несколько десятков слов. И все-таки с родным языком легче работать, чем с иностранным. А представьте себе, что русскому программисту пришлось «влезать» в операционную систему — там опять английский языковой барьер.

Но как бы то ни было, программист — это профессия, а у каждой профессии свои требования. У этой — знание английского. Хочешь работать программистом — учи английский. Ради удобства, ради унификации языкового компьютерного обеспечения можно на такое пойти.

Теперь рассмотрим языково-компьютерную проблему с позиций интересов пользователя, т. е. того, для которого программист составляет программу.

Если в паре компьютер—программист забота идет в основном об удобствах компьютера (а программист, мол, приспособится), то в паре пользователь — компьютер приходится заботиться о пользователе, чтобы ему было легко и удобно. В результате появляется еще одна группа «языков» или подпрограмм

— специально для облегчения диалога пользователь — компьютер. Например АОС — автоматические обучающие системы. Это как бы мостик от языка программирования к языку пользователя.

С англоязычным пользователем программисту полегче — разговор продолжается на английском. С русскоязычными пользователями ситуация иная, всех их не станешь заставлять учить английский. Значит, на последнем этапе нужно переходить к русскоязычному диалогу.

Проследим языковые перипетии в диалоге русского пользователя с компьютером.

Если даже компьютер оснащен операционной, трансляционной и автоматической обучающей системами, все равно пользователь сможет непосредственно выполнить на компьютере лишь простые операции. Чуть задача посложнее — нужно обращаться к программисту. Итак, пользователь излагает свою задачу программисту на русском языке. Программист переводит ее на английский язык программирования, скажем, на БЕЙСИК. Далее компьютер с помощью транслятора переводит эту программу на внутренний язык операционной системы и, наконец, непосредственно на «свой» язык машинных кодов. Только после этого программа выполняется. Ответ компьютера также проходит все эти стадии в обратном порядке. Жалкое зрелище представляет собой программа в этом случае: английские и русские слова вперемешку, да еще русские слова, написанные английскими буквами, и английские слова, написанные русскими буквами!

Кроме всего прочего, это еще и усложнение работы для русского программиста по сравнению с английским: ведь ошибку (или даже описку) в тексте на родном языке гораздо легче заметить, чем в таком англо-русском винегрете, какой представляет собой компьютерная программа для русскоязычного пользователя.

Пока сохраняется такой порядок работы, пользователя не очень волнует, на основе какого языка составлен этот самый БЕЙСИК. Однако сейчас одна из важнейших задач кибернетики — переход к общению с машиной на естественном языке, т. е. диалог с компьютером без программиста. Предполагается, что путь решения задачи таков. С одной стороны, язык программирования продолжает развиваться, все более приближаясь к естественному, а следовательно, программировать на нем будет все проще, поскольку программирование постепенно превращается просто в разговор с компьютером на обычном языке. С другой стороны, пользователь повышает свою компьютерную грамотность и поэтому легко овладевает языком программирования, ставшим почти естественным языком. Так программист и пользователь постепенно сливаются в одном лице.

Но... внимание! Такая схема годится лишь в том случае, когда язык программирования изначально построен на основе языка пользователя, т. е. в настоящее время лишь для англоязычного пользователя. А как же быть всем прочим?

Можно рассуждать так. Нужно отбросить национальные и политические амбиции и в интересах общечеловеческого дела ориентировать компьютерный язык на тот из человеческих, который завоевал наиболее твердые позиции в компьютеризации, который является базой для наиболее активных и наиболее распространенных компьютерных языков. А это — английский. Вот и нужно-де направить общие усилия на все большее приближение компьютерных языков к английскому, пока они не сольются с ним. И тогда все будет прекрасно: машины будут говорить на любом из человеческих языков, но компьютерные программы для них станут составляться только на английском.

Допустим, что так и произошло. Это значит, что в англоязычных странах в эпоху, когда общение с компьютером будет идти на естественном языке, любой пользователь компьютером будет в то же время и программистом, а в странах других языков доступ к программированию получат только те, кто знает английский.

В таком случае, может быть, выбрать хоть и человеческий, но «ничейный» язык из числа искусственных, например, эсперанто? Чтобы никому не давать «режима наибольшего благоприятствования». Да только здесь мы ничего не выиграем. Язык программирования таковым и останется (только его роль будет играть эсперанто), от естественных языков компьютер так и будет отгороженным, между компьютером и человеком сохранится каста жрецов-программистов. Нет, это не выход.

Попробуем подойти к проблеме с другой стороны. Не окажется ли так, что среди естественных языков будет выделяться лидер, а необходимость в едином компьютерном языке поможет закреплению доминирующих позиций лидера? И вот постепенно какой-то из естественных языков при поддержке основанного на нем компьютерного языка будет распространяться все шире, другие языки мало-помалу отойдут в тень, а потом и совсем исчезнут. И это окажется не обязательно английский. Кто знает, какие причины в этом процессе сыграют решающую роль. Возможно, распространенность языка в мире — и тогда есть шансы у английского; возможно, число говорящих на языке людей планеты — и тогда нет конкурентов у миллиардного китайского; а может быть, и удобство программирования на данном языке — тогда повышаются шансы у японского и опять-таки китайского: ведь их иероглифы — готовые операторы.

Значит, все снова свелось к старому вопросу о том, сформируется ли на Земле единый общечеловеческий язык. Наверное, при благоприятном развитии человеческой цивилизации будет когда-нибудь у всех людей единый язык на основе ли какого-то национального или как некий «обобщенный», вобравший в себя все лучшее из других языков. Как знать?!

Но в ближайшее время и в обозримом будущем ожидать такого не приходится. Пока что, напротив, с ростом национального самосознания, с развитием наций укрепляются и развиваются национальные языки, а тенденций к их всеобщему слиянию что-то не заметно.

Развитие машинных языков не сталкивается с национальным своеобразием естественных языков лишь потому, что любой язык компьютера крайне примитивен и по сравнению с любым человеческим языком просто убог. А как только язык машины начнет ощутимо приближаться к языку человека, национально-языковой проблемы не избежать.

Многие кибернетики убеждены, что язык — это лишь система условных коммуникативных знаков. Для машинного языка это действительно так. Но любой естественный язык принципиально не подходит под такое определение. Если единицы языка и считать знаками, то совсем особыми — мотивированными, теснейшим образом связанными с мышлением, с психикой человека. Человеческий язык — выразитель мышления, чувствований, культуры, мировоззрения и вообще всего развития и бытия говорящего на нем народа. Язык создается всей жизнью народа, вбирает в себя эту жизнь и выражает ее.

Эти принципиальные различия машинных и естественных языков сказываются и на их устройстве.

В любом языке, хоть в компьютерном, хоть в человеческом, главное — смысловая сторона или семантика. В машинном языке семантика одноплановa, одномерна. Каждый знак в нем обладает лишь логическим, понятийным значением. Но единицы естественного языка (например, слова) обладают многоплановой, многоаспектной семантикой. В ней логико-понятийное значение хотя и главный, но не единственный аспект, как ядро в живой клетке или в атоме. Мало того, и самих этих ядер у одного слова может быть несколько (полисемия: прыгает кошка, прыгает мяч, прыгает давление, прыгает изображение на экране телевизора); и у сходных по содержанию ядер может быть различная форма (синонимия: учитель — преподаватель — педагог), или, напротив, при тождестве форм совершенно разные содержания (омонимия: ключ — «родник»; ключ — «отмычка»); и кроме всего этого, вокруг ядра еще клубятся семантические ореолы.

Сравним, например, слова «бабушка», «бабка» и «старуха». Толковый словарь так определяет их понятийное значение (т. е. логико-понятийное ядро): «бабушка» — то же, что бабка; «бабка» — старая женщина, старуха; «старуха» — женщина, достигшая старости. Как видим, понятийные ядра у этих трех слов одинаковы. Но можно ли в обращении заменить слово бабушка словом старуха? Не советую вам это делать, особенно если в руках у женщины, достигшей старости, скажем, зонтик.

Каждый говорящий по-русски, понимает, что эти слова оценочно, качественно различны. Можно сказать, что при сходстве понятийных ядер семантика этих слов резко различается качественными ореолами. Этот аспект семантики имеется у каждого слова. Иногда он более плотен, более заметен, осознан, в других случаях почти не замечается, но в языковом подсознании присутствует всегда.

Более далек от ядра, довольно разрежен и в основном подсознателен звукосодержательный ореол. Он создается содержательностью самого звучания, выразительностью звуковой формы слов. Можно подумать, что это маловажный аспект семантики, о котором не стоит здесь рассуждать. Но оказывается, он существенно влияет на жизнь и функционирование слова. Вот, к примеру, «лютик». Понятийно слово первоначально обозначало то же, что и злючка (лютик от слова лютый, т. е. злой). Но название этого ядовитого (злого) цветка звучит очень красиво, нежно. Поэтому его качественный ореол под влиянием звукосодержательного стал из отрицательного положительным. Такое «давление» звучания на значение действует, конечно, и сейчас, сдвигая восприятие семантики слов в ту или иную сторону.

Звукосмысловые силовые линии формируют семантические группы слов. Синонимы выстраиваются в ряды

ЯЗЫКИ программирования

ФОРТРАН (FORTRAN) — один из первых языков программирования, языков долгожителей, широко применяемый и сегодня, несмотря на критику и мрачные прогнозы. Название его составлено из первых букв FORmula TRANslating Syst., что означает «система трансляций формул». Язык был разработан группой сотрудников фирмы IBM под руководством Джона Бэкуса. Первый отчет по созданию ФОРТРАНА (версия ФОРТРАН I) был опубликован 10 ноября 1954 г. (последний датируется 1957 г.). Первая версия была реализована на компьютере IBM-704. Для написания первого компилятора потребовалось 18 человеко-лет.

В первые годы язык был встречен скептически, так как ожидали снижения эффективности программ. Однако Д. Бэкус с сотрудниками группы создали такой компилятор, который генерировал программы, по качеству не уступающие составленным вручную на машинном языке.

Язык получил всеобщее признание, в особенности в области научных и инженерных вычислений, для этих целей он используется и поныне.

К характерным чертам языка относится сравнительная простота и легкость его изучения, близость записи арифметических выражений к обычной математической записи, возможность работы с комплексными переменными и переменными двойной точности, удобные и развитые операторы ввода-вывода данных. К достоинствам относят также простоту компилятора, эффективность получаемых объектных программ, возможность независимой компиляции подпрограмм и относительную легкость в обнаружении ошибок в программах.

Наличие многочисленных диалектов не создало серьезных препятствий для широкого обмена программами.

Первая версия языка имела некоторые недостатки и ограничения при ее использовании (например, имена переменных не должны были превышать шесть литер) и представляла лишь ограниченные возможности для определения структур данных. Серьезные недостатки были в управляющих структурах. Точки ветвлений приходилось определять метками и следить за тем, чтобы смысл программы не терялся в множестве операторов безусловного перехода.

Небезынтересно отметить, что и дальнейшие версии несут в себе отпечаток ФОРТРАНА I и его реализации на ламповой ЭВМ IBM-704. В частности, шестилитерные имена переменных объясняются тем, что длина слова в IBM-704 составляла 64

не только по значению, но и по звучанию. Противопоставление смыслов в антонимичных парах часто поддерживается противопоставлением звучаний.

К тому же слова естественного языка связаны еще и ассоциациями. Так, слово «фрукт» у русского вызывает ассоциацию — яблоко, «глаз» — видеть и т. д.

А ведь все это говорилось лишь о лексическом значении слова. Есть еще и грамматическое значение (рода, числа, падежа и т. д.). А далее предложение, текст, где семантика многократно усложняется.

Кибернетики, пожалуй, скажут:

— Да зачем компьютеру все эти семантические сложности. Для него главное — логико-понятийное значение, а все прочее — это эмоции, тонкости, нюансы; оставим их человеку.

На мой взгляд, это то же самое, как если бы вытащить из живой клетки ядро и пытаться заставить его функционировать, а потом пробовать собрать из таких «голых» ядер живой организм. Даже и случись каким-то образом такое, полученный «ядерный» организм все равно не смог бы функционировать так, как клеточный.

То же и в естественном языке — семантика не может быть только ядерной, аспекты значения слиты, сплавлены воедино и функционируют все вместе, сложно взаимодействуя друг с другом.

Что ж, — вздохнет кибернетик, — если так, то, хотя это и усложняет дело, придется обучать компьютер всем аспектам семантики, только и всего.

Да, задача в принципе разрешима, и кое-какие результаты в этом направлении уже имеются. Компьютер делает первые шаги в освоении понятийного ядра, постепенно постигает качественный ореол, неплохо справляется со звукосодержательным, обучается ассоциативному значению, разным аспектам синтаксической семантики.

Но дело как раз в том, что с продви-

ФОРТРАН

бит. Максимальное число индексов для элементов массива, равное 3, объясняется наличием в IBM-704 только трех индекс-регистров и т. д.

В 1958 г. появилась усовершенствованная версия языка — ФОРТРАН II, а спустя несколько лет — ФОРТРАН III, которая, однако, не получила распространения.

Наиболее популярной получилась версия под названием ФОРТРАН IV, созданная в 1962 г. В этом же году Американский национальный институт стандартов (ANSI) организовал комитет по выработке стандарта ФОРТРАНа, который был принят в 1966 г. Вновь разработанный стандарт получил название ФОРТРАН ANSI или ФОРТРАН-66.

После 1966 г. было разработано еще несколько версий ФОРТРАНА, на основе которых в 1977 г. появилась пересмотренная версия стандартного языка под названием ФОРТРАН-77. Эта версия не исключает использования старых фортрановских программ, она лишь расширяет возможности языка в операциях ввода-вывода, в средствах описания данных и подпрограмм, в конструкциях, которые ранее допускали значения только целого типа, и т. п.

В мае 1984 г. комитет X3.13 Американского национального института стандартов рассмотрел предложения по выработке новой стандартной версии языка ФОРТРАН на период 1980—1990 гг. под названием ФОРТРАН 8x. В новой версии, совместимой с языком ФОРТРАН-77, предлагаются изменения, касающиеся в основном работы с файлами: процедур открытия и закрытия файлов, доступа к ним и поиска записи в файле с помощью различных операторов, влияния операторов на эффективность работы процессора.

Трансляторы языка ФОРТРАН существуют на всех вычислительных машинах и, в частности, на машинах серии СМ и диалого-вычислительных комплексов (ДВК).

В СССР в 1978 г. разработаны стандарты на языки ФОРТРАН (ГОСТ 23056—78) и базисный ФОРТРАН (ГОСТ 23057—78). Стандарт ФОРТРАН ANSI положен в основу реализации версии ФОРТРАН IV на ЕС ЭВМ.

Литература

Брич З. С. и др. ФОРТРАН ЕС ЭВМ. — М.: Статистика, 1978.

Грунд Ф. Программирование на языке ФОРТРАН IV. — М.: Мир, 1976.

Колдербэнк В. Программирование на ФОРТРАНЕ: ФОРТРАН-66 и ФОРТРАН-77. — М.: Радио и связь, 1986.

Хелмс Г. Л. Языки программирования / Пер. с англ. — М.: Радио и связь, 1985.

М. П. МАЛЫХИНА,
А. П. ЧАСТИКОВ

жением компьютера «в глубь» значения, с освоением им все новых аспектов семантики его язык будет становиться все более национальным.

Даже понятийно-ядерная (казалось бы, общелогическая) семантика любого естественного языка отражает национальное мировосприятие народа. В свое время лингвистами, психологами, философами бурно обсуждалась так называемая гипотеза лингвистической относительности Сепира—Уорфа, но, кажется, никто из обсуждавших никак не связывал языковые факты, приводимые Э. Сепиром и Б. Л. Уорфом, с проблемой единого компьютерного языка. Но вот, оказывается, такая связь есть.

Напомним вкратце гипотезу. Центральным ее моментом является утверждение о том, что действительность отражается в сознании человека, пройдя через преломляющую призму языка. Такое утверждение, как кажется, ведет к агностицизму, поскольку получается, что мы воспринимаем мир не таким, каков он есть, а таким, каким его представляет нам язык. «Выпрыгнуть» из собственного языка, чтобы посмотреть на действительность «со стороны», помимо языка, как будто бы невозможно, значит, мы и не знаем, какова действительность на самом деле, а вынуждены принимать ее в искаженном призмой языка виде.

В застойный для нашей философской науки период считалось достаточным доказательством неправомерности фактов уже одно подозрение в том, что они «льют воду не на ту мельницу». Поэтому языковые факты гипотезы лингвистической относительности нашей наукой внимательно не рассматривались, а сама гипотеза была официально отвергнута как идеалистическая.

Но давайте разберемся. Отрицать, что язык сильнейшим образом влияет на формирование сознательной картины мира, просто нелепо. Как же иначе ребенок постигает действительность? Если бы он делал это только непосредственно с помощью органов чувств, то он навсегда остался бы в развитии на уровне животного. Очевидно, что человеческая, интеллектуальная картина мира в его сознании формируется в основном с помощью языка, под воздействием речи окружающих его людей, затем средств массовой информации, книг. А какую картину действительности могут сформировать в сознании ребенка взрослые? Только ту, которая есть в их сознании, другой у них нет. И что же, эта картина мира объективна, не искажена, точно соответствует реальной действительности? Вся история отдельных народов и всего человечества свидетельствует, что это не так.

Когда-то люди верили, что в доме живет домовой, в лесу бродит леший, а в реке плещется русалка. Ребенок того времени получал такую картину мира только с языком взрослых — ведь ясно, что с помощью органов чувств он ее сформировать не мог. Люди того времени считали эту картину мира действительной, реальной, правильной, но теперь-то мы знаем, как она была искажена.

А возьмите религиозные картины мира. Они формируются лишь языком и ничем больше, но как они развиты, всеобъемлющи, красочны! Они создают труднопреодолимое впечатление реальности. Недаром для многих верующих иллюзорный мир их религии более реален, чем действительный. Подумать только, человек под воздействием искаженной языковой религиозной картины мира может отказаться от реальной жизни и даже принести себя в жертву религии, т. е., по сути дела, религиозному языку! Сила воздействия языка на сознание колоссальна, а во времена господства религиозного мировоззрения язык религии пронизывал собой весь язык народа.

Вы спросите, какое отношение все это имеет к машинным языкам? Дело в том, что приведенные примеры — лишь наиболее яркое и очевидное свидетельство своеобразия языковых картин мира. И сейчас каждый национальный язык рисует в какой-то мере свою картину действительности.

Так, еще совсем недавно русский язык (а не реальная действительность) формировал такую картину мира, где

присутствовал «корифей всех времен и народов», где люди умирали «за Сталина», где «поет и пляшет вся Советская страна», процветает «развитой социализм», а «нынешнее поколение людей будет жить при коммунизме». Такая картина действительно многим (слишком многим) казалась реальной. Сейчас видно, как она была искажена, но можно ли считать, что уж вот теперь-то все в нашем миропонимании правильно и никаких искажений больше нет. Разумеется, это было бы крайне наивно. Мы просто не знаем этих искажений, и только наши потомки будут поражаться странной аберрации нашего мировосприятия, как мы поражаемся искаженному сознанию предка, приносящего себя в жертву божеству, или голодающего народа, умиленно возносящего хвалу «великому кормчему».

В оправдание себе можем лишь быть абсолютно уверены в том, что и в сознании потомков будут свои искажения, и неизвестно еще, лучше ли наших. Для нас же важно, что все эти преломления действительности в сознании непременно отражаются в национальном языке и с языком передаются каждому новому поколению. Родители, конечно, не злодеи, чтобы с помощью языка намеренно формировать в сознании ребенка искаженное представление о действительности. Просто они не знают, что в их собственном миропонимании правильно, а что нет. Поэтому стараются передать все. Компьютер — тоже дитя человеческое, и, обучая его языку, мы не сможем отделить зерна от плевел и научим его только тому, что знаем сами. А это значит, что каждый национальный язык принесет с собой в «сознание» компьютера свою картину мира со своими искажениями. Как же узнать, какая из картин лучше? И согласится ли какой-либо народ, чтобы в его сознание вместе с компьютерным языком внедряли чужую картину мира?

Значит ли все это, что национальные своеобразия языков и сознаний непреодолимы? Или что действительность принципиально непостижима, поскольку каждый язык преломляет ее по-своему? Отнюдь нет.

Счастье человечества — в непослушании детей. Если бы дети были абсолютно послушными, если бы они полностью доверяли воспитателям, покорно усваивали бы их картину мира и никогда ничего в ней не изменяли, то их мышление в точности повторило бы мышление воспитателей, и интеллектуальное развитие человечества прекратилось бы.

Родители упорно и настойчиво стремятся всеми способами вложить свое миропонимание в сознание нового поколения, что объяснимо и оправданно — ведь опыт предшествующих поколений огромен и ценен, без него невозможно интеллектуальное становление нового поколения. Потому-то и прилагает общество колоссальные усилия по обучению и воспитанию нового поколения (т. е. по формированию в их сознании своей картины мира), используя все средства — от колыбельной песни до развитых систем образования, привлекая к делу литературу, искусство, агитацию, пропаганду, средства массовой информации, а в случае непослушания, инакомыслия пуская в ход наказания от отцовского ремня до мощного государственного репрессивного аппарата, который не останавливается даже перед физическим уничтожением непокорных.

И все же (не поразительно ли это!) даже малый ребенок не доверяет полностью той картине действительности, которую рисует ему с помощью языка взрослый. Он постоянно проверяет эту картину и собственным опытом, и опытом окружающих.

— Не трогай, обожжешься, — говорит мама малышу.

Казалось бы, он должен безоговорочно поверить ее опыту. Но нет — он тянет палец к горячей плите.

Вот великий дар природы интеллекту человека — дар непослушания, сомнения, поиска!

Конечно, не все так прямолинейно. Полная анархия и пренебрежение родительским опытом тоже ни к чему хорошему не приведет. Отказ от колоссального интеллектуального наследия предков немедленно отбросит человечество в пещеры. Но и дублирование их мышления без проверки и дальнейшего

развития ведет к застою, к остановке прогресса. Родительский опыт показывает: если ребенок чересчур непослушен и не желает учиться — это прямой путь к деградации и хулиганству; если же он излишне вышколен и забит — значит, из него вырастет безынициативный и нетворческий обыватель. Здесь нужны разумные пропорции, и соблюсти их помогает только опыт каждого человека и всего человечества.

— Не ходи на реку — русалка утащит, — пугает дед шустрого внука. А тот, хоть и побаивается, но лезет в воду и даже ныряет, во все глаза высматривая под водой русалку. Ее, однако, не видно, не обнаруживают коварную девицу с рыбьим хвостом и сотоварищи внука.

Важнейший момент! Именно здесь разрушаются агностические философские страхи перед гипотезой лингвистической относительности: в том случае, когда опыт не подтверждает языковой картины мира, выбор рано или поздно делается не в пользу языка, а в пользу реальной действительности.

Все дело в диалектике. Если рассматривать язык, мышление и действительность в статике, тогда да: мышление воспринимает действительность через преломляющую призму языка. Если же, как требует диалектика, все эти явления рассматривать в динамике, то языковая картина мира постоянно сравнивается с действительностью через опыт, в результате чего искажения в языковой картине исправляются, и она все больше приближается к действительности. Конечно, с постижением новых сторон действительности могут возникать новые искажения в языковой картине этих сторон действительности. Но таков уж путь человеческого познания, включающий не только объективные знания о действительности, но и ошибки, заблуждения мышления, фиксирующиеся в языке и передающиеся потом на проверку опыту, который преодолевает ошибки, обеспечивая интеллектуальный прогресс человечества. И так всегда.

Следы преодоленных заблуждений остаются в языке, как остались в языке слова «леший», «домовой», «русалка», «ангел», «черт», или, скажем, «совнархоз». Но теперь эти слова не искажают наше мировосприятие, а остаются лишь как метки истории народа, как элементы национального своеобразия языка.

Рассуждая о гипотезе лингвистической относительности, мы говорим пока только о понятийной семантике языка. Она хорошо осознаваема, и поэтому мышление преодолевает языковые искажения в этой области сознательно, рационально, в том числе и с помощью науки. Но как только мы вступаем в область семантических ореолов, дело многократно усложняется, поскольку это либо полусознательные, либо целиком подсознательные аспекты семантики. А ведь явления лингвистической относительности касаются и их.

Возьмем, к примеру, грамматическое значение рода. В русском языке оно есть, в английском его нет. В русской грамматике это следы древнего искаженного антропоморфного восприятия мира, когда всем предметам и явлениям приписывались человеческие свойства, в том числе и различия по полу. Сейчас, разумеется, это искажение преодолено. Мы, русские, отнюдь не считаем стол мужчиной, а скамью женщиной. Но это на сознательном уровне. А можно ли считать, что род существительных не оказывает на наше языковое «полусознание» вообще никакого влияния?

Попробуйте перевести на английский язык песню «Что стоишь, качаясь, тонкая рябина». Адекватно сделать это невозможно, потому что весь смысл песни построен на том, что слово рябина — женского рода, а дуб — мужского. В английском языке оба эти слова «никакого» рода, поэтому пришлось бы объяснять, что рябина — девушка, а дуб — мужчина, т. е. выводить семантику на логический уровень. Но это было бы понимание, а не чувствование; очарование полусознательного постижения немедленно исчезнет, останется банальное «лобовое» уподобление.

Еще более наглядный пример — строчки А. Вознесенского:

Ты кричишь, что я твой изувер,
и, от ненависти хорошея,
изгибаешь, как дерзкая зверь,
голубой позвоночник и шею.

Перевести это на «безродовой» английский язык вообще невозможно, даже понятийно. То, что «ты» — это женщина, подсказано «грамматическому полусознанию» единственным способом — женским родом слова «дерзкая». Но это прилагательное неправильно согласуется с существительным мужского рода зверь, что, однако, допускается нашим языковым восприятием, потому что существительные такой формы могут иметь и женский род (например, дверь). Эффект такой «родовой игры» невозможно описать, его можно только почувствовать, причем только тому, кто говорит на «родовом» языке.

Да и весь русский язык буквально пронизан родовой грамматической семантикой. Олицетворение березки, ивы, сокола, орла производится в русском языке не путем сравнений или уподоблений, а естественно вытекает из самой грамматической формы слов. Если о человеке сказано «старый пень», то русскому человеку не нужно объяснять, что это сказано о мужчине, а если «выдра», то о женщине. Хотя есть выдры самцы, грамматическая форма запрещает иные истолкования, кроме выдра — женщина. На безродовой язык переводы в таких случаях невозможны, а добавочное указание будет не переводом, а истолкованием, что уже совсем не то.

Вот и получается, что хотя сознанием языковой антропоморфизм преодолен, русское семантико-грамматическое «полусознание» особым образом окрашивает наше мировосприятие, придавая языку специфический национальный колорит. И это касается, конечно, не только категории рода. Все грамматическое устройство языка порождает национально-языковой семантический ореол, вносящий свой вклад в языковое сознание народа.

Проявляется лингвистическая относительность и в области качественного ореола языковой семантики. Одни и те же понятийные ядра слов зачастую получают в разных языках различные качественные оценки. Скажем, в языках Крайнего Севера понятие «собака» оценивается только положительно, а в языках Ближнего и Среднего Востока это слово часто используется по отношению к человеку в бранном смысле. По-русски выражение «коровьи глаза» звучит обидно, а в индийских языках это комплимент. Если русский назовет женщину лошадью или кобылой, то это грубое оскорбление, а по-арабски — похвала.

Качественный ореол в разных языках плохо изучен, но те, кто им занимается, говорят, что почти каждое слово имеет в разных языках более или менее резко различающиеся качественные признаки.

Специфичны в разных языках и ассоциативные связи слов. Ясно, что слово «зима» в русском языковом сознании вызовет совсем иные ассоциации, чем соответствующее слово, скажем, в испанском. Для русского с этим понятием будут связаны такие слова, как «мороз», «снег», «санки», «вьюга», «метель», «сугроб», чего, понятно, нельзя ожидать для языкового сознания испанца.

Да что там зима! Вот уж, казалось бы, понятие «отец» у носителей любого языка должно бы вызвать одинаковые ассоциации. Ничего подобного. У русских и американцев с этим понятием связаны такие слова, как «мать», «семья», «наставник», «учитель». Но вдруг у американцев появляется странная для нас ассоциация — деньги, семейный бюджет. А ничего странного. Если в современной русской семье держателем и распорядителем семейного бюджета является чаще всего мать, то в американской — обычно отец.

Звукосодержательный ореол, как и все другие аспекты семантики, тоже имеет национально-языковую специфику. По результатам специальных психометрических экспериментов можно сказать, что для русских звук «хь» самый неприятный, некрасивый (хищный, хитрый), а в восприятии немцев это вполне приятный звук. В немецком языке сходный звук встречается гораздо чаще, чем «хь» в русском, а

потому для немцев он обычен, привычен и оценивается довольно хорошо. Поляки все шипящие звуки оценивают лучше, чем русские, что опять-таки связано с высокой частотой шипящих в польской речи.

Если оценки всех русских звуков речи, выраженные в числе, занести в память компьютера, то можно составить программу, по которой компьютер сможет определить (вычислить) выразительность (содержательность) звучания слов и текстов.

Оказывается, что для большинства звукосодержательных слов наблюдается интересная закономерность — их звучание находится в соответствии со значением. Скажем, слово «любовь» по оценке компьютера звучит как нечто «хорошее, нежное, женственное, красивое, яркое, доброе», а хулиган как «плохое, грубое, темное, устрашающее и низменное».

Есть, хотя и в меньшем количестве, и слова с противоречиями между звуковой формой и содержанием. Например, слово «юноша» звучит женственно, а жуткое чудовище названо «хорошо и светло» звучащим словом «василиск». Таким словам живется неуютно в языке, и они либо меняют свое значение, как это произошло со словом «лютик», либо уходят в пассивный запас, как уже давно случилось со словом «василиск».

В стихотворениях многих талантливых поэтов также часто наблюдается гармония звучания и содержания, которую способен выявить компьютер. Анализируя лишь звуковую ткань текста, компьютер сообщает, что в стихотворении Пушкина «Зимнее утро» говорится о чем-то ярком, нежном и радостном, а в стихотворении «Зимний вечер» — о чем-то темном, печальном, тоскливом. «Я помню, любимая, помню» Есенина звучит как нечто прекрасное и нежное, а «Левый марш» Маяковского — как стремительное и сильное.

Более того, звуки речи связаны в нашем подсознании еще и с цветом, что тоже чувствуется талантливыми поэтами и используется в их творчестве.

Поразительный пример. Арс. Тарковский написал стихотворение «Пускай меня простит Винсент Ван Гог» под впечатлением от картины художника «Звезда и кипарис». Просчитанный на компьютере звукоцвет стихотворения дал ту же основную цветовую гамму, что и на картине!

И это не единственный случай. То же самое наблюдается в стихотворении Блока по картине В. Васнецова «Гамаюн, птица вещая», в стихотворении Арс. Тарковского «Петровские казни» по картине Сурикова «Утро стрелецкой казни» и в других.

Здесь не мистика, а еще одно свидетельство глубоко национального своеобразия языковой семантики, своеобразия, пронизывающего все мельчайшие поры языка, вплоть до содержательности его звуков.

Итак, формирование единого общечеловеческого языка — это формирование общечеловеческой языковой семантики во всех ее аспектах. Чтобы сформировалась единая понятийная семантика, мало создать общий толковый словарь, нужно, чтобы выработалось общечеловеческое мировоззрение и миропонимание. Для этого придется объединить многие аспекты национальных историй и культур.

Чтобы сформировался единый качественный ореол, нужно свести воедино закрепленные в языках национальные оценки.

Единый язык должен также выработать единые ассоциативные связи слов, единые грамматические значения.

Наконец, формирование общей звукосодержательности потребует выработки единых подсознательных реакций на сами звуки речи, что затрагивает глубинные корни национальной языковой семантики.

Сколько нужно времени, чтобы прошли все эти процессы, невозможно сказать, ясно только, что очень много. Единый язык человечества возникнет никак не раньше, чем прекратятся войны, исчезнут границы и отдельные государства, начнут формироваться общая экономика и культура.

А до тех пор пока общечеловеческого языка не будет, не сможет функционировать и единый высокоразвитый, приближающийся к естественному

компьютерный язык. Пожалуй, в ближайшее время, пока компьютерные языки еще крайне примитивны, будет наблюдаться стремление кибернетиков создать единый язык программирования, хотя и сейчас ориентация компьютерных языков на разные цели уже создает центростремительные силы, отдаляющие разные версии языков программирования друг от друга.

С приближением же машинных языков к естественным компьютерные языки все заметнее будут пропитываться национальной языковой спецификой и начнется все большее их обособление. Когда будет осуществлен речевой ввод и вывод информации (что уже делается), когда компьютер будет общаться с человеком без программиста, он заговорит не на некоем усредненном машинном языке, а на русском, английском, японском, китайском и т. д.

Другое дело, что машинная переработка языковой информации, речевое управление механизмами, машинный перевод и развитие других приложений киберлингвистики смогут помочь формированию единого человеческого языка, но все-таки это дело весьма отдаленного будущего.

Сейчас, похоже, просматриваются некоторые новые технические возможности. В компьютерах пятого поколения разработчики надеются обойтись вовсе без операционных систем, трансляторов и даже без специальных языков программирования, «встроив» все языковые функции непосредственно в «железо» компьютера. Тогда компьютер будет понимать человеческую речь (естественно, на национальных языках), а смена языка обеспечится сменой электронных плат машины.

А на кибернетическом горизонте маячат уже нейроноподобные машины, которые в самом устройстве будут имитировать человеческий мозг...

Так что вопрос о языке программирования может отпасть сам собой.

Но при любых технических достижениях проблемы национальных особенностей языка в общении с компьютером не отпадут до тех пор, пока существуют национальные языки.

В наших рассуждениях показаны лишь немногие глубинные языковые семантические трудности, с которыми столкнется кибернетика при обучении компьютера естественным языкам. Многое, очень многое осталось незатронутым. Но сейчас кибернетикам, пожалуй, не особенно интересно читать даже и то, что обсуждалось, потому что это проблемы не сегодняшнего и даже не завтрашнего дня, однако чем раньше мы их осознаем, чем лучше осмыслим, тем успешнее они будут решаться, когда такая необходимость возникнет.

П. П. ВАСИЛЬЕВ,
С. А. ФРОЛОВ,
С. И. МАЛЫШЕВ. Мы живем в удивительное время. Человечество вступает в новую техническую эру — эру вычислительных машин и роботов, наделенных искусственным интеллектом. Уже сегодня вычислительные машины управляют полетом космического корабля и работой химического цеха, помогают в медицине и продают билеты на транспорт, исследуют психику человека и играют в шахматы, сочиняют музыку и проектируют еще более современные ЭВМ. Не так далек тот день, когда вычислительная техника буквально окружит нас, создаст такую среду для жизни и деятельности человека, в которой проблема общения человек—машина и машина—машина станет одной из самых актуальных.

Человеку придется научиться говорить с самыми разными «умными» машинами и заставить их понимать друг друга. Поэтому весьма естественным кажется вопрос о необходимости создания некоего универсального машинного языка. Но можно ли вообще создать универсальный машинный язык? Если можно, то почему он до сих пор не существует? Если нельзя, то почему?

Услышав подобные вопросы, люди, даже малознакомые с вычислительной техникой, среагируют по-разному. Оптимисты воскликнут: «Как, уже?! То, что еще совсем недавно было собственностью фантастов, обсуждается специалистами? Блестяще! Какими темпами развивается наука!» Пессимисты резонно возразят: «Опять морочите

нам голову очередной универсальной панацеей. Да, мы не знакомы с техническими возможностями вычислительной техники, но... были уже «универсальные» средства от тараканов и от облысения. Добавьте к любому понятию слово «универсальный», и вы сразу придете к явному абсурду. Примеры? Сколько угодно. Универсальное средство передвижения — гибрид велосипеда, космического корабля, подводной лодки и дирижабля? Универсальная одежда — это то, в чем можно ходить и на пляже, и на арктической станции, и в театре? Правда, есть универсальные магазины, но и им до полной универсальности далековато».

Однако большинство, видимо, просто решит: «Об этом говорят специалисты, значит, тут что-то есть. Пока все это не очень понятно, но... Давайте рассмотрим все по порядку».

Согласно ГОСТ 19781—83 машинным языком называется язык программирования, предназначенный для представления программы обработки данных. Добавка «универсальный» должна, по-видимому, предполагать такую форму представления программы, которая обеспечит ее выполнение на любых средствах обработки данных, т. е. на любых ЭВМ.

Поэтому сначала нужно разобраться в том, как машина понимает и выполняет задание для работы — программу.

Вычислительная машина, точнее ее центральный исполнительный орган — процессор или микропроцессор, последовательно выполняет некоторые элементарные действия по обработке информации. Набор этих элементарных действий специфичен для конкретного типа процессора, он выбирается заранее при разработке процессора из соображений удобства использования и эффективности функционирования процессора. Различные наборы для различных процессоров включают от 100 до 1000 простейших действий, т. е. машинных операций. Каждая машинная операция однозначно кодируется некоторым двоичным числом, называемым машинной командой, а вся совокупность машинных кодов образует машинный язык конкретного процессора. Процессор понимает и исполняет только последовательность машинных команд (программу) и ничего больше.

Значит, огромная армия программистов во всем мире только и делает, что пишет последовательности чисел — машинных команд? И машинный язык является средством общения программиста и ЭВМ? Нет, так было только в самом начале развития вычислительной техники. Теперь подавляющее большинство программистов понятия не имеют о том, какое, например, число кодирует простейшую операцию пересылки числа из одной ячейки памяти в другую.

Дело в том, что программисты вынужденно оказались, пожалуй, самыми прогрессивными сторонниками автоматизации по сравнению с представителями других отраслей техники. Практически сразу они поняли, что писать программы в виде чисел неблагодарное, трудоемкое, а главное, очень неэффективное дело, и разработали для себя

систему автоматизации проектирования программ (да-да, тот самый САПР, который только сейчас, спустя три десятилетия, входит в арсенал конструкторов других отраслей). Сначала было решено заменить числа машинного языка удобными для восприятия и использования мнемоническими инструкциями (ПЕРЕСЛАТЬ, СЛОЖИТЬ, СРАВНИТЬ и т. д.) — появился язык АССЕМБЛЕР, затем родились более сложные и удобные языки: ФОРТРАН (для научных расчетов), КОБОЛ (для экономистов) и другие. И это еще в пятидесятых годах!

Все эти языки получили общее название языков программирования, предназначенных для представления программ. Но как ни удобны языки программирования для программистов, ЭВМ по-прежнему понимает только свой родной машинный язык. Поэтому каждый язык программирования оказался разбитым на две части: первая, не зависящая от типа ЭВМ, — собственно язык, как совокупность удобных программисту средств и правил записи последовательности действий ЭВМ; вторая, зависящая от типа ЭВМ, — переводчик (его назвали транслятором), преобразующий предложения языка программирования в конкретный машинный язык. Заметим, что транслятор — это программа, которую обычно пишут на АССЕМБЛЕРе — языке, близком к машинному, и только транслятор с АССЕМБЛЕРа (и только он!) действительно пишется на машинном языке. Забегая вперед, скажем, что уже здесь проявилась тенденция к универсализации языка программирования: программист может написать одну программу на ФОРТРАНе, пригодную для выполнения на различных типах ЭВМ (если для каждой из этих ЭВМ существует транслятор с ФОРТРАНа).

Описанная терминология принята в среде специалистов по вычислительной технике, она конкретна и не вызывает неоднозначных толкований.

Говоря более обобщенно, под машинным языком можно понимать всю совокупность средств, обеспечивающих взаимодействие человека и ЭВМ на различных этапах разработки и эксплуатации программ, т. е. все то, что машина «понимает» прямо (собственно машинный язык) или с помощью трансляторов (языки программирования), а также в процессе выполнения конкретной пользовательской программы, например команды бухгалтера при расчете на ЭВМ заработной платы.

Поэтому возможность создания универсального машинного языка следует рассматривать на нескольких уровнях (слоях).

Во-первых, возможно рассмотреть тенденции к универсализации на уровне машинных ходов и АССЕМБЛЕРа, пытаясь, образно говоря, создать универсальный язык общения компьютеров.

Во-вторых, следует рассмотреть уровень общения человек — машина и попытаться создать универсальный машинно-независимый язык программирования.

Первый путь потребует, очевидно, создания универсальной аппаратуры ЭВМ, т. е. сведет сложную задачу к еще более сложной, поскольку не ясно, что такое универсальная аппаратура. Второй путь представляется более реальным.

Но прежде чем углубляться в его исследования, постараемся понять, зачем нужен универсальный машинный язык? Может быть, это просто фикция, не имеющая реальных корней? Корни есть, и питают их реальные экономические и психологические потребности общества.

В ходе естественного прогресса науки и техники, в частности вычислительной техники, исторически сложилось так, что в нашей стране, да и во всем мире, существуют многие типы ЭВМ, отличающиеся друг от друга внутренним машинным языком. Существует (почему, мы обсудим позднее) и огромное число различных языков программирования — несколько сотен или, по некоторым оценкам, несколько тысяч. Для каждого языка программирования реализованы трансляторы, «привязанные» к конкретному типу ЭВМ, причем обычно каждый транслятор понимает свой, несколько отличающийся от других диалект языка про-

граммирования. Даже для одного типа ЭВМ различные формы из конкурентных соображений реализуют свои собственные трансляторы (и следовательно, диалекты) для одного и того же языка программирования. Например, для ЭВМ IBM PC существует более 20 различных трансляторов (диалектов) языка СИ, и каждый имеет несколько совместимых сверху вниз, но все-таки в чем-то отличных версий. Итого несколько (десятков) тысяч диалектов языков программирования. Что за бессмыслица, это же вавилонское столпотворение языков. Но, увы, это реальность. Огромные средства тратятся на разработку, распространение, сопровождение, изучение различных языков и программ на них. Не проще ли создать один универсальный язык программирования, обучить ему всех программистов и писать программы только на нем? Это обеспечило бы огромную экономию затрат на разработку языков, решило бы проблему переноса разработанных программ с одной ЭВМ на другую, позволило бы использовать накопленный при разработке программ опыт, облегчило бы изучение программирования и использование ЭВМ (проблема компьютерной грамотности) и т. д. В конечном итоге все это привело бы к резкому ускорению научно-технического прогресса в целом. Такие доводы — главный аргумент в пользу создания универсального языка программирования. Сюда же следует добавить психологические особенности человека: изучив один язык программирования, он обычно неохотно изучает другой, третий и т. д. Масло в огонь подливают и фантастические темпы развития вычислительной техники: то, что еще десяток лет назад казалось недостижимой мечтой, сегодня выпускается серийно, порождая надежду на реализацию в будущем еще более смелых проектов.

Тенденция универсализации стала просматриваться еще на начальном этапе появления языков программирования и их трансляторов для различных типов ЭВМ. Приведенные ранее доводы настолько очевидны, что были сформулированы специалистами еще в начале 60-х годов. Тогда же была сделана первая попытка реализации универсального языка-оболочки ПЛ/1 на основе лучших черт существовавших к тому времени языков ФОРТРАН, АЛГОЛ, КОБОЛ. ПЛ/1 — хороший язык, им до сих пор пользуются программисты, но универсальный язык не состоялся, попытка оказалась неудачной. Идея универсализации была похоронена? Нет. После анализа причин поражения в середине 70-х годов была сделана новая попытка. Были разработаны требования к языку, проведен международный конкурс, в результате появился язык АДА. Удача? Снова нет. Язык АДА только начал использоваться на практике, а уже ясны его некоторые недостатки как универсального языка. Будут ли сделаны следующие попытки? Безусловно.

Возникает вопрос: а можно ли вообще создать универсальный язык? Может быть, что-то мешает его появлению, кроме недостатка технических знаний и опыта?

Скорее не мешает, а противостоит. Тенденции универсализации противостоит диалектически связанная с ней тенденция к специализации, проявления которой мы ежеминутно встречаем во всем многообразии жизни. И, двигаясь по пути специализации, общество имеет возможность получить и реально получает те же выгоды, тот же огромный экономический эффект, который обещает универсализация.

Основная заповедь экономики — максимальные результаты при минимальных затратах — приводит к естественному выводу: в языке программирования не должно быть ничего лишнего, не связанного с конкретной задачей. Но разрабатывать под каждую задачу свой язык тоже дорого. А вот для решения целого класса похожих задач создание специального языка вполне приемлемо.

Проявление тенденции к специализации машинных языков имеет ряд объективных причин.

Во-первых, исторически сложилось несколько конкурирующих направлений в области разработки аппаратных средств и программного обеспечения

вычислительной техники. Примером такой конкуренции является соперничество двух ведущих фирм США — IBM и DEC.

Чьи решения лучше? Кто победит? На этот вопрос пока нет ответа. А техническая реализация ЭВМ накладывает свою специфику и на машинный язык.

Во-вторых, объективно существуют различные категории пользователей как по потребностям, так и по подготовке и профессиональным интересам. Одних машинный язык интересует лишь в той степени, которая позволила бы решать несложные вычислительные, логические, информационно-поисковые задачи, так сказать, общего назначения. Другим язык нужен как инструмент профессиональной деятельности, в которой используется особый профессиональный язык. И здесь говорить об универсальности становится практически невозможно. Язык профессионального юриста сильно отличается от языка химика, а понять, о чем говорят физики-теоретики на своей конференции, например по сильным взаимодействиям, смогут вообще немногие.

Далее, ограничения технические и связанные с условиями работы требуют от машинных языков различных возможностей. Это требования по объему программы, быстродействию, по интерфейсу и др. Они так или иначе сказываются на структуре языков и приводят к их дифференциации. Любой универсальный язык окажется слишком избыточным для конкретной вычислительной машины.

Кроме того, когда говорят об универсальном машинном языке, как правило, подразумевают только один этап жизненного цикла программного средства — кодирования программы. Другие важные этапы — алгоритмизация, отладка, тестирование, документирование, освоение пользователями и сопровождение требуют наличия специфических конструкций языков, применяемых на данных этапах.

Идее универсального языка во многом противоречит и практический опыт программирования. Даже программисты, которые продолжают работать на ПЛ / 1, используют обычно лишь определенное подмножество языка, крайне редко прибегая к непривычным для себя конструкциям, стараясь избежать их в срочных и ответственных программах.

Как видно, тенденция к специализации имеет очевидные права на существование, и время показывает, что число машинных языков (а это связано в первую очередь с их специализацией) продолжает расти. Список хорошо известных языков АЛГОЛ, ФОРТРАН, КОБОЛ, БЕЙСИК, ПЛ / 1 пополняется новыми популярными названиями — ПАСКАЛЬ, ЛИСП, ПРОЛОГ, СИ, МОДУЛА-2, АДА. Число только весьма широко распространенных языков давно уже перевалило за 500.

Быстрый прогресс в области технологии РЭА, особенно в области БИС, позволил сформулировать еще одну тенденцию, в значительной степени определяющую интерес к специализации. Все чаще делаются попытки разрабатывать не язык (а следовательно, и транслятор) для конкретной существующей ЭВМ, а наоборот! Сначала разрабатывается язык для решения каких-то конкретных прикладных задач, а затем разрабатывается специализированная ЭВМ, которая наиболее эффективно реализует данный язык. При этом, естественно, архитектура специализированных машин может быть совсем не похожа на архитектуру так называемых универсальных машин.

До сих пор мы рассматривали только противоречия двух тенденций универсализации и специализации. Но диалектика учит, что борьба противоположностей не существует без их единства, взаимного перехода и взаимного преобразования.

Задуманный в качестве универсального, язык ПЛ / 1 стал удобным языком решения задач в конкретном, но достаточно широком диапазоне применений на ЭВМ практически одного типа (IBM 360 / 370, ЕС-ЭВМ), т. е. в некотором смысле специализированным языком. «Универсальный» язык АДА также займет подобающее место в ряду специализированных языков, в частности, как язык разработки больших программных

проектов и в качестве инструментального средства программирования встроенных специализированных ЭВМ.

С другой стороны, язык СИ, задуманный как специализированный язык системного программирования, оказался «встроенным» в операционную систему (большая часть ОС была написана на СИ), вместе с ней получил широкое распространение. Затем он был реализован для других ОС и в конечном итоге стал универсальным (со значительными оговорками, конечно) языком программирования в широком диапазоне применения, оттесняя с занятых позиций языки ПАСКАЛЬ, ПЛ / 1, ФОРТРАН, АССЕМБЛЕР.

Специализированный язык логического программирования ПРОЛОГ неожиданно нашел применение в сфере искусственного интеллекта и был принят в качестве базового языка для разрабатываемой в Японии системы ЭВМ пятого поколения.

Таким образом, в ходе технического прогресса, первоначально разрабатываемые как универсальные, языки становятся специализированными, и наоборот, разрабатываемые как узкоспециализированные, они порой становятся универсальными для какой-то прикладной области.

Рассмотрим еще один аспект данной проблемы. Если не удается сделать универсальным язык нижних уровней (язык команд, АССЕМБЛЕР), алгоритмический язык высокого уровня, язык специфический, то, может быть, естественный язык станет универсальным и разрешит все проблемы? К сожалению, и естественный язык (если его удастся реализовать) не дает надежды на универсальность. Все названные тенденции к специализации сохраняются, поскольку они существуют вместе и параллельно с естественным языком. По всей видимости, появится класс машин, которые будут понимать естественный язык. Но возможность создания достаточно сложной программы на естественном языке представляется весьма проблематичной. Видимо, и естественный язык останется одним из специальных языков, на котором будут взаимодействовать человек и ЭВМ.

Какой вывод можно сделать на основе приведенных рассуждений? По всей вероятности, смысл вопроса о возможности создания универсального машинного языка можно понимать как отражение некоторой тенденции, имеющей место в диалектической борьбе двух противоположностей — универсального и специфического, общего и частного. И разрешение этого извечного противоречия может быть только в техническом и экономическом компромиссе, достижимом в конкретный исторический момент.

А. Б. РОДИОНОВ. Существующее на настоящий день обилие алгоритмических языков, используемых для программирования компьютеров, может смутить любого начинающего программиста, пытающегося выбрать из них тот, который подходит ему больше всего для решения данной конкретной задачи. Советы более опытных коллег могут зачастую лишь внести дополнительную путаницу, так как может оказаться, что один из них любит пользоваться компилятором с языка ПАСКАЛЬ, другой — пишет только на С, а третий вообще никогда не знал ничего другого, кроме интерпретатора с языка БЕЙСИК. Если же коллеги начинающего программиста работают на достаточно мощных компьютерах, то ему могут посоветовать писать все свои программы на таких языках, как ПРОЛОГ или АДА, программисты «старой закалки» могут порекомендовать АЛГОЛ, ФОРТРАН, КОБОЛ, ПЛ / Л или даже RPG. Найдутся и горячие сторонники языков АПЛ и ЛИСП, так же как и отдельные энтузиасты других, менее распространенных языков или языковых диалектов... Чаще всего каждый из них будет утверждать, что для данной задачи лучше всего подходит тот язык и тот компилятор (или интерпретатор), которым пользуется он сам и к которому он больше всего привык.

Какой же язык выбрать? И вообще, зачем нам так много языков? Не лучше ли придумать один, Универсальный Язык Для Программирования Любых

Классов Задач? Можно учить этому языку детей в школах (и заодно навести порядок в курсе школьной информатики). Можно обеспечить компиляторами с этого языка все существующие типы ЭВМ (единовременное мероприятие такого рода обойдется не так уж дорого) и обязать разработчиков снабжать все новые модели компьютеров компиляторами с этого языка.

Как следствие «всеобщая компьютерная грамотность» получит благоприятную почву для дальнейшего развития и расцвета, программы с компьютеров одного типа можно будет легко переносить на компьютеры другого типа, не надо будет учить новые языки программирования только из-за того, что на новом компьютере нет компилятора с уже знакомого языка...

Блестящие перспективы! Остается проверить, первыми ли мы до этого додумались? К сожалению, нет. История уже знает попытки создания языков программирования, претендующих на роль универсальных, — все они потерпели неудачу. Рассмотрим причины этих неудач на двух наиболее известных примерах: языках ПЛ / 1 и АЛГОЛ-68.

Язык ПЛ / 1 был разработан фирмой IBM для компьютеров IBM / 360 (позднее IBM / 370) и предполагался для использования в качестве «универсального языка программирования» как системными, так и прикладными программистами. Важной особенностью ПЛ / 1 было также то, что полный вариант языка был реализован (в отличие от языка АЛГОЛ-68, полный вариант которого не был реализован вообще).

Тем не менее, несмотря на мощность языка и наличие нескольких хороших компиляторов с ПЛ / 1, предполагаемый всеобщий переход программистов на ПЛ / 1 так и остался несбывшейся мечтой его разработчиков. Прикладные программисты, конечно же, использовали новые компиляторы с ПЛ / 1, но продолжали также пользоваться и компиляторами с языком ФОРТРАН, КОБОЛ, RPG, а системные программисты писали свои программы на языке АССЕМБЛЕР (в значительной степени это определялось тем, что АССЕМБЛЕР / 360 обладал одним из самых мощных макрогенераторов) и лишь сравнительно небольшая группа энтузиастов работала исключительно на ПЛ / 1.

Оказалось, что желание «угодить на любой вкус» при разработке ПЛ / 1 пагубно сказалось на самом языке. Его конструкции «утяжелились», обилие средств и операций породило множество громоздких операторов — язык не приобрел такой желанной для программистов «простоты» при достаточной выразительной силе. И конечно же, не могло быть и речи о том, чтобы перенести компилятор с такого языка на машины с другой архитектурой из-за слишком большой «привязанности» его к архитектуре IBM / 360 / 370 и операционным системам этой корпорации. Справедливости ради отметим, что такие попытки все же предпринимались, но терпели неудачу.

Здесь же заметим, что язык С, разработанный гораздо позже для машин PDP-11 Д. Ритчи и не претендующий на роль «универсального языка для всех», а создаваемый как язык системного программирования для операционной системы UNIX, сегодня приобретает все большее число сторонников среди системных программистов, работающих на самых разных компьютерах (к их числу относится и автор этих строк). Это связано в первую очередь с простотой и четкостью языка, а также его большой выразительной силой. Однако этот язык не может быть рекомендован для начинающих, так как он требует известных знаний в области системного программирования и имеет некоторые скрытые особенности и даже противоречия.

На приведенных выше примерах мы можем видеть, что существуют как минимум два класса задач, требующих различных языковых выразительных средств: это класс задач системного программирования (программы для разработки других программ) и класс прикладных задач (программы для получения результатов в других областях науки, техники, искусства и пр.). Первый из них обычно требует доступа к ресурсам

компьютера и операционной среды для управления этими ресурсами, тогда как второй ориентирован только на уже существующие стандартные системные возможности. Совмещение различных средств этих двух классов в рамках одного языка может оказаться крайне затруднительным и привести к созданию громоздкого «мертвого» языка, даже при успешной разработке высокоэффективных и надежных компиляторов.

Язык АЛГОЛ-68 постигла еще более незавидная участь, чем ПЛ / 1. Этот язык, задуманный не просто как «язык для всех программистов», но и как «язык для всех машин», так и не был полностью реализован ни на одном комьютере. Несмотря на то что АЛГОЛ-68 был гораздо более стройным с концептуальной точки зрения, чем ПЛ / 1, он требовал очень громоздких компиляторов в силу чрезмерного обилия средств, предлагаемых программисту, и сверхуниверсальности в подходе к языку как к объекту разработки (достаточно заметить, что в языке присутствовали даже элементы программы-генератора компиляторов!). Можно сказать, что АЛГОЛ-68 погубила его универсальность — он предназначался для всех сразу и ни для кого в отдельности.

До сих пор мы рассматривали два основных класса задач: задачи системного программирования и задачи проблемного программирования. Но не следует забывать, что внутри этих классов существуют свои подклассы задач, сильно отличающиеся друг от друга и, следовательно, налагающие определенную специфику на разрабатываемые для них языки. В частности, в области системного программирования методы и средства, используемые для разработки, например, компиляторов, кардинально отличаются от методов и средств, используемых для разработки, к примеру, графического интерфейса системы. А в области прикладного программирования разработка системы заказов железнодорожных билетов не имеет практически ничего общего с системой для расчета траекторий небесных тел или с системой, предназначенной для автоматизации труда композитора. Приведенные в качестве примеров задачи описываются в совершенно различных терминологических плоскостях и, следовательно, требуют в идеале для их компактного описания своих, специализированных языков программирования. Можно возразить, что тем не менее такие задачи можно решать средствами какого-либо одного универсального языка программирования, мирясь с его неприспособленностью применительно к каждому отдельно взятому классу задач и неудобствами, из этой неприспособленности проистекающими. Но если развить эту мысль несколько дальше, то любую из этих задач можно в принципе решить и с помощью машины Тьюринга (отвлекаясь от ее быстродействия)! Речь здесь идет не о принципиальной возможности реализации какого-либо алгоритма на компьютере (этот вопрос достаточно разработан в теории конечных автоматов), а скорее об оптимизации использования человеческого времени, необходимого для программирования задач различных классов (хотя, конечно, возможны компромиссы на уровне специальных библиотек прикладных программ и / или макросов).

Таким образом, наш Универсальный Язык Для Программирования Любых Классов Задач начинает приобретать все большую бесплотность из-за удручающего обилия Классов Задач и их терминологической и операционной несовместимости друг с другом. Однако еще не все потеряно!

Особым классом языков, имеющих право претендовать на универсальность, являются расширяемые языки, т. е. такие, выразительные средства которых могут расширяться за счет доопределения пользователем новых конструкций языка и, таким образом, настраиваться на новые классы задач. К ним относится в первую очередь язык ФОРТ, а также языки ЛИСП, РЕФАЛ и некоторые другие. Но доопределение синтаксиса, и тем более семантики языка, требует серьезных знаний в области системного программирования и не может быть рекомендовано для всех пользователей без исключения, тем

более что базовые версии таких языков обычно включают в себя относительно небогатый исходный набор языковых примитивов. Таким образом, этот класс языков также не подходит для использования в качестве универсального.

Итак, Универсальный Язык Для Программирования Любых Классов Задач не может быть создан? Как ни странно это может показаться на первый взгляд, но такой язык уже существует! Это наш, человеческий язык общения друг с другом — язык содержательной постановки задач.

Направление, в котором может осуществляться дальнейшее развитие методов программирования, лежит в рамках области под названием «искусственный интеллект», которая быстро развивается. Задачи, относящиеся к этой области, требуют гигантских объемов оперативной и долговременной памяти и одновременной работы многих высокопроизводительных процессоров. Тем не менее автор не теряет надежды, что уже при его жизни мощная экспертная система сможет в процессе диалога с человеком определить класс решаемой задачи, информационная система — запросить и уточнить исходные данные (в том числе, пользуясь базами знаний, доступными по линиям связи), блок порождения и оценки алгоритмов — оценить возможности и пути решения задачи (продемонстрировав их предварительно человеку), а генератор программ — создать и выполнить соответствующую программу. Автор также надеется, что в результате всей этой работы сможет появиться, как минимум, статья, подобная той, которую вы только что прочитали.

С. С. ЛАВРОВ. Авторы представленных в этом сборнике статей почти единодушны в своем мнении: универсальный машинный язык — это заманчиво, но нереально. И только А. Ф. Дедков и А. Л. Щерс высказываются в конце своей статьи чуть осторожнее: «Может ли быть в конце концов создан действительно универсальный язык, мы не знаем».

Мне, как программисту, подход к проблеме, развиваемый П. П. Васильевым, С. А. Фроловым и С. И. Малышевым, представляется наиболее объективным, обоснованным, полным и относящимся к сути дела. Да, универсальный машинный язык ни сейчас, ни в обозримом будущем не реален. Но все же...

Доводы, обосновывающие эту точку зрения (а они во многом повторяются от статьи к статье), можно было бы и пополнить. Например, любая программа для ЭВМ сама порождает свой собственный язык, и даже несколько. Она требует от пользователя подготовки и ввода исходных данных. Это делается обязательно на некотором языке. Иногда этот язык примитивен — нужно просто заготовить последовательность числовых значений некоторых величин в определенном порядке или «ткнуть пальцем» (подвести курсор) в определенное место экрана и нажать кнопку. Но некоторые программы, например тестовые редакторы, требуют от пользователя гораздо более развитых средств общения с ними и владения определенной системой понятий, понимания того, что происходит внутри машины во время ее работы. Ответные сообщения программы, результаты ее работы тоже выводятся на некотором языке, требующем определенных знаний и усилий для их понимания.

Итак, кроме языков для записи программ существуют языки общения с самими программами, а они, несмотря на определенные тенденции, каноны и традиции в их проектировании и использовании, оказываются еще более разнообразными и нестандартными. Это усугубляет хаос в общении человека с компьютерами.

Но есть некоторые факторы и тенденции, действующие в противоположном направлении. Некоторые из них лишь обозначены в представленных здесь статьях, а некоторые вовсе не упомянуты.

А. П. Журавлев ярко и выразительно говорит о трудностях общения между людьми разных национальностей, воспитанных в духе своего национального языка, на базе своей национальной культуры. Но как ни трудно, люди-то общаются. Статьи и книги переводятся

с языка на язык, переводится не только научно-техническая, но и художественная проза и даже стихи.

Поэт-англичанин, конечно, сумеет перевести на английский цитируемые А. П. Журавлевым строчки А. Вознесенского. Кроме тривиального he-beast есть масса других способов передать мужскую природу слова *«зверь»* и противоречивость словосочетания «дерзкая зверь». Гораздо сложнее, кстати, для восприятия слова «голубой позвоночник». Стихи нельзя переводить «понятийно», но переводить их все-таки можно.

Проблема здесь не языковая. Речь идет о взаимодействии, общении и обогащении различных национальных культур. Разные народы живут на одной Земле. Чтобы жизнь эта была спокойнее, насыщеннее и интереснее, надо изучать чужие языки и чужую культуру. А смешение языков (французского с нижегородским) — проблема, порожденная отнюдь не компьютерами и не ФОРТРАНом.

Суть дела не в сложностях межнационального общения и взаимопонимания, а в ложности критерия — «компьютерный язык тем лучше, чем он ближе к естественному языку».

Два программиста (или физика-теоретика, или лингвиста), скажем, русский и японец, лучше поймут друг друга, изъясняясь на ломаном английском, чем англичанин-биолог и англичанин-геолог, если последние попытаются общаться на научные темы, а не об архитектурных красотах Ленинграда или удобствах ленинградских гостиниц.

Проблема решения задач на ЭВМ — это общечеловеческая, а не национальная проблема. Ее нужно осознать, найти к ней подходы, выделить ее составные части, разработать методы их решения. Как лучше распределять машинную память, если ее не хватает? Хорошо ли пользоваться операторами go to или лучше обходиться без них? Как организовать бесступиковое взаимодействие параллельно выполняемых процессов? Разные люди будут по-разному отвечать на подобные вопросы, но не потому, что их родные языки различны, а потому, что различен их опыт решения задач на ЭВМ, различны машины, на которых они работали (здесь еще можно усмотреть влияние национальных условий), и классы задач, с которыми они сталкивались.

Проникновение в суть подобных вопросов — это тоже овладение определенной культурой, хотя и не в гуманитарном, а в научно-техническом понимании. Общность этой культуры создает гораздо более существенные условия для взаимопонимания, для плодотворного общения, а значит, — и для выработки единого языка, чем общность национальных языков.

У вычислительной техники в разных странах свои, более или менее единые законы развития и трудности роста. Совместные усилия инженеров и ученых многих стран помогают преодолевать эти трудности и направлять развитие по прогрессивному в целом пути. Вот это единство проблем и усилий по их решению и объединяет разработчиков машин, программ и языков вокруг этих проблем, служит основой для выработки универсальных решений и подходов. А различие опыта, условий и традиций порождает разнообразие и специфику подходов. Но без такого разнообразия и специфики, как отмечают все авторы сборника, невозможны ни выработка единых (хотя бы в основе) решений, ни их оценка.

Итак, несомненно существующее единство наиболее общих проблем в развитии вычислительной техники и программирования толкает исследователей на поиск универсального решения этих проблем. И если средство от облысения когда-нибудь будет найдено, то оно, скорее всего, окажется универсальным от Исландии до Мозамбика.

Обратимся еще раз к проблемам естественного языка, теперь для людей одной национальности. Я могу во многом не соглашаться со взглядами А. П. Журавлева на проблемы компьютерных языков. Я могу также протестовать, когда П. П. Васильев и его соавторы пишут, что машины что-то «проектируют», «сочиняют» и даже «исследуют» (между прочим, не что-то, а психику человека!). Но при всем этом я

могу проследить и понять основную мысль авторов сборника (хотя и упуская некоторые частности) и именно потому, что все мы говорим на одном естественном языке. Я — один из носителей этого языка. В моем понимании и употреблении многих его слов и оборотов много индивидуального, специфического, возможно, даже ошибочного. Но любого человека, говорящего по-русски, я могу выслушать, расспросить, попросить разъяснить мне слова, которыми он пользуется, и смысл, который он в них вкладывает, и в конце концов как-то понять. Если собеседник человек далекой от меня специальности, то на такое понимание могут уйти месяцы и годы, но в принципе оно возможно. Все это потому, что мы говорим на одном, едином и универсальном для нас, родном языке. Окажись я на одном необитаемом острове с человеком, владеющим лишь суахили, и ситуация оказалась бы в корне иной. Здесь потребовались бы месяцы для налаживания общения на весьма простые темы, вроде ожидаемой погоды на завтра.

Необходимость иметь средство для быстрого установления контакта с машиной — важное обстоятельство, заставляющее программистов-системщиков интересоваться проблемами универсального языка программирования, оценивать языки по критерию их универсальности и тратить усилия на реализацию именно «универсальных», пусть в кавычках, языков.

П. П. Васильев и другие пишут: «Добавка «универсальный» должна, по-видимому, предполагать такую форму представления программы, которая обеспечит ее выполнение... на любых ЭВМ». Я бы здесь заменил слово «программа» оборотом «любая программа обработки любых данных». Универсальностью в этом понимании обладает каждый из более или менее распространенных языков.

Если мы владеем общими понятиями, методами и приемами программирования (а их можно излагать и изучать худо-бедно на примере лишь одного из таких языков), то написать программу решения любой задачи на любом таком языке почти одинаково просто (или сложно). Дело, как правило, не в языке, а в самой задаче. Вникнуть в ее суть, найти метод ее более или менее эффективного решения — вот главная трудность.

А освоить синтаксические тонкости того или иного алгоритмического языка — дело десятое. На это у хорошего программиста уйдет день-два, ну десять. Конечно, и через полгода он будет кое на чем еще спотыкаться, но гораздо больше камней преткновения доставляет обычно сама задача, особенно если она из непривычной проблемной области.

В этом смысле любой из широко известных языков универсален. Конечно, для вычислительных задач ФОРТРАН или АЛГОЛ удобнее ЛИСПА, а для формульных преобразований — наоборот, но это не противоречит сделанному утверждению. Конечно, на языке, вам хорошо знакомом, программировать легче, но случись вам перейти на машину, где этот язык не реализован, обойдетесь и другим языком — нужда заставит.

Проблема универсальности имеет еще одну сторону — единство операционной среды. Несколько удивительно, но именно лингвист А. П. Журавлев, а не программисты — авторы сборника, отметил, что пользователь обычно общается с машиной на нескольких языках — языке собственно программирования, языке управляющих карт или директив операционной системы, языке диалога при работе с терминала, языке текстового редактора.

Но вот явление последнего десятилетия — на многих персональных ЭВМ невысокого класса («домашних», вроде нашей БК-0010) пользователю доступен чаще всего лишь БЕЙСИК, успешно совмещающий все эти функции. Вы ощущаете, конечно, некоторую разницу между его командами и операторами, но те и другие вам одинаково просто доступны. Операторы вы обычно включаете в состав программы, а команда подается непосредственно, но в большинстве случаев можно поступить и наоборот. Ваши действия по редактированию программы почти не отличаются от ее первоначального ввода. Вся

работа идет в диалоге и ничего, кроме БЕЙСИКА, вам при этом не нужно.

Такое слияние, сращивание исходно различных языков создает удивительный комфорт для пользователя. Осознание этого комфорта поставило в повестку дня создание «интегрированных» систем программирования или шире — систем работы на ЭВМ. Термин не из удачных, но идея понятна и привлекательна. Войдя в такую систему с утра, вы можете не выходить из нее целый день, и все ваши нужды будут удовлетворяться. Элементы подобной «интеграции» в удовлетворении потребностей пользователя сейчас проникают во многие системы.

Среди чисто отечественных разработок подобную интеграцию средств работы на ЭВМ можно отметить в языке Эль-76 многопроцессорного вычислительного комплекса «Эльбрус». Она позволила написать все общесистемное программное обеспечение (без всякого изъятия, без применения машинно-ориентированных языков «низкого» уровня) этого комплекса и значительную часть прикладных программ только на этом языке. В этом отношении язык Эль-76 гораздо более универсален, чем ФОРТРАН, АЛГОЛ-68 или ПЛ/1, хотя и не отличается от них по уровню.

Еще один подход к универсальности — это так называемые языки без нижнего уровня. Поясним идею на примере естественного языка. Ребенок рано узнает слово «складывать». Ему понятны фразы: «Сложи игрушки», «Сложи свою одежду», «Сложи кубики» (по картинке). Однако слова «Сложи эти числа» — для него пустые звуки. Со временем и они наполняются содержанием (правда, несколько иным), то есть создается необходимый нижний уровень. В школе и в вузе он узнает о возможности складывать (по-разному, хотя в чем-то и похоже) не только числа различных классов, но и многочлены, векторы, матрицы и т. д. — нижний уровень обогащается и расщепляется.

Использование языка без нижнего уровня напоминает программирование сверху вниз. Но если традиционно, спускаясь сверху вниз, мы ориентируемся на определенный набор средств нижнего уровня и, дойдя до этого уровня, останавливаемся, то здесь остановиться можно в принципе в любом месте. Программирование на таком языке помогает нам уяснить задачу и подход к ее решению, не доводя программу до реализации. Когда мы достигли уровня, на котором структура решения уже вполне ясна, можно переписать программу на другом, традиционном языке, сохраняя эту структуру. Но можно и продолжить раскрытие верхних уровней, ориентируясь на любые средства исполнения программы, например на определенный машинный язык. Этот язык можно выбрать, когда структура решения определилась и определила какие-то требования к языку реализации программы. А можно с какого-то места разветвить процесс раскрытия уровней и получить несколько вариантов программы на разных языках.

У меня нет личного опыта работы с подобными языками, и я лишь пересказываю то впечатление, которое вынес, слушая рассказ о языке CDL-2.

На многоуровневость языков можно взглянуть и по-другому. Как только мы создали и даже пока еще создаем некоторый, алгоритмический или иной, язык, нам нужно иметь средство для описания этого языка, его синтаксиса и семантики, а также явлений и закономерностей, наблюдаемых при пользовании этим языком. Такое средство принято называть метаязыком (я допустил некоторую вольность в определении содержания этого термина). Над метаязыком можно надстроить метаметаязык и т. д. Все это обычно обрывается на естественном, родном языке человека, который достаточно успешно выполняет функции метаязыка не только для всех искусственных языков, но и для иностранных языков и даже для самого себя.

Вот это последнее свойство — способность языка описывать свое собственное устройство, смысл и функционирование — может служить еще одним критерием универсальности. Среди программистов носителем такого свой-

ства прославился язык ЛИСП, на котором сразу же после или даже одновременно с появлением самого языка был описан его интерпретатор — правила выполнения программ на этом языке.

Впоследствии этот прием — составление транслятора для некоторого языка на самом этом языке — применялся неоднократно и получил наименование раскрутки. Способность к такому самоописанию тоже служит признаком или характеристикой универсальности языка. Впрочем, если язык пригоден для записи любых программ обработки любых данных, то почему программа-транслятор должна быть исключением?

Можно услышать, что в классе проблемно-ориентированных языков не может быть универсального языка. В связи с этим напрашивается замечание очень общего характера. Если нет универсального языка, то, может быть, существует универсальный метаязык, то есть универсальное средство создания проблемно-ориентированных языков.

Действительно, в мире средств общения человека с машиной возникли так называемые языки представления знаний. Если традиционные алгоритмические языки оперируют величинами (их именами) и их значениями, то в языках представления знаний важнейшие объекты — это понятия (термины), их свойства и взаимосвязи. Идут споры, есть ли здесь принципиальная разница. Может быть, большой разницы и нет, но даже терминологическое различие символично. Языки для работы со знаниями уже существуют, применяются и причем довольно успешно. Имея такой язык, можно его средствами строить модели (понятийные) различных проблемных областей и, следовательно, создавать различные проблемно-ориентированные языки. Таков еще один подход к универсальности.

Для самых отважных несколько слов о математической стороне дела. Когда в 30-х годах математики вплотную взялись за уточнение понятия «алгоритм», почти одновременно возникло несколько формализаций этого понятия. Наиболее известны машины Тьюринга нормальные алгорифмы* Маркова, лямбда-исчисление Черча, но есть и другие. Оказалось, что все эти формализации в известном смысле эквивалентны. А именно, любая из них позволяет написать алгоритм, переводящий алгоритмы с языка другой формализации в свой собственный или в третий язык. Я бы не стал этому удивляться, потому что подобная универсальность является естественным свойством хорошей формализации понятия алгоритма. Впрочем, сейчас, 50 лет спустя, легко об этом рассуждать.

Любой из ходовых алгоритмических языков позволяет составить программу, моделирующую работу любой машины Тьюринга, или программу, исполняющую любой нормальный алгорифм Маркова, и т. д. В этом смысле любой такой язык весьма универсален. Одно только «но». Машины Тьюринга и др. — это машины с бесконечной, точнее, неограниченно расширяемой памятью. А память любой ЭВМ может быть увеличена, даже за счет устройств внешней памяти, лишь до определенных пределов. Это означает, что универсальность абстрактных математических «машин» принципиально отличается от универсальности реальных компьютеров и языков программирования. Модель машины Тьюринга на самом современном суперкомпьютере может остановиться из-за исчерпания памяти там, где сама машина Тьюринга еще работала бы и работала. Впрочем, работающая на чердаке вычислительного центра машина Тьюринга — это из серии первоапрельских шуток.

Итак, понятие универсальности само не универсально, имеет множество граней и смысловых оттенков. Тем не менее сама универсальность (в определенных практических рамках) живет, развивается и приносит в программировании немалую пользу, о чем я и пытался рассказать.

---

* Я пишу «алгорифм» из уважения к светлой памяти А. А. Маркова, именно так писавшего это слово.

ПИСЬМО ЧИТАТЕЛЯ

О. Н. ГРИГОРОВ,
кандидат технических наук,
старший научный сотрудник,
доцент, г. Москва

Прошло уже более четверти века с тех пор, как я связал свою жизнь с вычислительной техникой — с вопросами применения электронных вычислительных машин в самых различных областях.

На моих глазах происходило рождение и становление отечественной вычислительной техники: ЭВМ семейства «Урал», машины Минского завода, ЭВМ «Весна», «Снег», «Стрела», семейство машин БЭСМ, М-20, М-220, «Днепр», «Мир» и др.

Я был свидетелем перехода от машин первого поколения к машинам второго поколения, но, несмотря на смену поколений, неизменным оставалось то, что отечественные машины второго поколения, как и машины первого поколения, разрабатывались советскими конструкторами.

Их математическое обеспечение (операционные системы, системы автоматизации программирования) было написано на русском языке, что делало его понятным для широкого круга пользователей и не вызывало трудностей при изучении и последующем использовании.

При возникновении у пользователей ЭВМ (разработчиков систем, программистов) каких-либо сложных ситуаций можно было всегда получить исчерпывающую консультацию у конструкторов-разработчиков ЭВМ и математического обеспечения, которые (что весьма естественно для разработчиков) знали свое детище, как говорится, вдоль и поперек. Зачастую на основании замечаний разработчиков систем и эксплуатационщиков производились доработки, в значительной степени улучшающие характеристики машин.

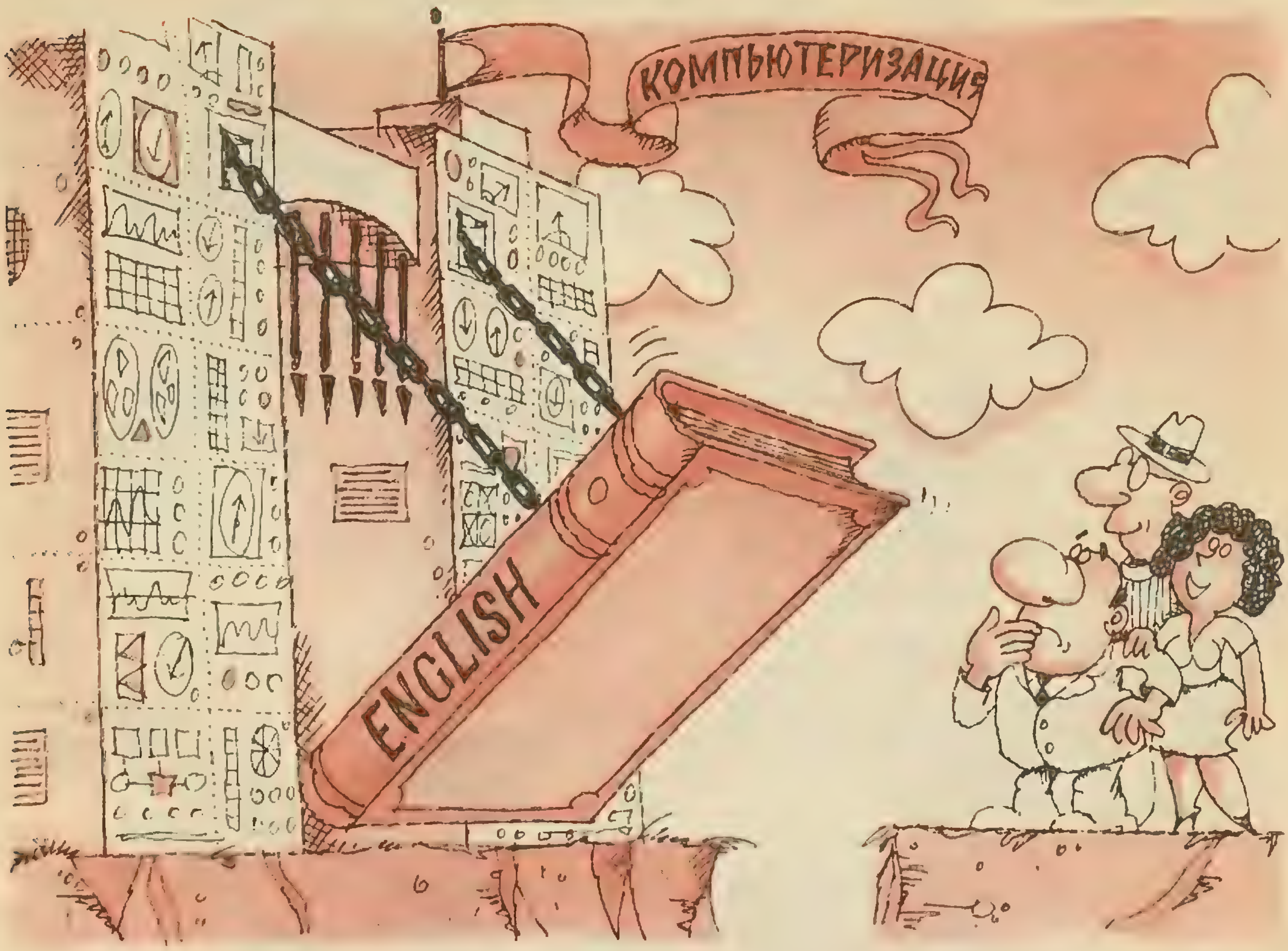
Имена конструкторов вычислительных машин (Лебедев, Рамеев, Бурцев, Карцев и др.) и возглавляемые ими коллективы были известны в стране и пользовались глубоким уважением.

В конце 60-х — начале 70-х годов техническая политика в области средств вычислительной техники совершила крутой поворот: было принято решение не о разработке отечественных конструкций ЭВМ 3-го поколения, а о подражании зарубежным машинам 10—15-летней давности, таким, как IBM-360, PDP9(11) Хьюллет Паккард. Отечественные аналоги указанных ЭВМ получили наименование ЕС-ЭВМ, СМ-ЭВМ (СМ-3(4), СМ-1(2) соответственно).

Начались трудности разработки и освоения новых машин. Этот процесс в части освоения, или, как теперь говорят, адаптации (заметьте, адаптации, а не разработки), математического обеспечения операционных систем и систем автоматизации программирования затянулся до сегодняшних дней и конца ему не видно.

В значительной степени возросли трудности в освоении математического обеспечения разработчиками систем управления, программистами в основном из-за отсутствия качественной технической документации, а также из-за того, что весь диалог человека с ЭВМ стал осуществляться не на русском языке, а на английском. К тому же не просто на английском, а на техническом жаргоне. И еще хуже — не просто на жаргоне, а вдобавок ко всему на его аббревиатурах.

Резко, в 4—5 раз, возросли сроки подготовки специалистов-программистов, операторов ЭВМ. Повысились требования к уровню квалификации специалистов по эксплуатации и применению вычислительной техники. До сих пор нет хороших описаний математического обеспечения, да и не может быть! Как можно хорошо описать то, чего сам не делал, а у того, кто делал, не проконсультируешься! Все это, вместе взятое, привело к резкому снижению полезного времени использования ЭВМ. Такая ситуация длится уже полтора десятка лет, и конца ей не видно.



Чтобы лучше понять, к чему уже привела и приведет в дальнейшем существующая техническая политика в создании ЭВМ, основанная на зарубежных аналогах, представим себе следующую ситуацию. Допустим, что наши полиграфисты захотят создать печатную машину на основе какого-либо даже самого прекрасного зарубежного образца и при этом скопируют не только саму машину, но и иностранный шрифт. Ведь тогда все желающие приобщиться к печатной продукции должны будут выучить иностранный язык.

Может быть, и не следовало бы приводить подобных примеров, но сегодня в стране разворачиваются работы, ориентированные на выпуск сотен тысяч компьютеров в год, оснащение ими средних школ, ПТУ, техникумов. Но все разработки, предполагаемые к запуску в производство, даже школьные компьютеры, «говорят» все на том же английском языке.

Количество вывезенных из-за рубежа операционных систем и их версий, алгоритмических языков и их версий выражается двухзначным числом. Наличие большого разнообразия операционных систем и алгоритмических языков на западе определяется конкуренцией фирм разработчиков ЭВМ и математического обеспечения и их стремлением к захвату рынков сбыта, но совершенно не обосновано в условиях нашего государства.

В своем выступлении в г. Тольятти товарищ М. С. Горбачев поставил задачу разработки и выпуска продукции, превышающей по своему уровню лучшие зарубежные образцы. Очень хотелось бы услышать, как собираются выполнять эту установку нашей партии товарищи из ГКВТИ, ученые — родоначальники создания в нашей стране англоязычной вычислительной техники и активные продолжатели этой пагубной линии. Парадоксально, но факт: все сегодняшние ЭВМ не имеют советских

конструкторов, которые бы несли ответственность за технический уровень выпускаемой вычислительной техники. Сегодня все претензии следует предъявлять к конструкторам иностранных фирм, с которых в данном случае, как говорится, взятки гладки. Наши специалисты могут нести ответственность в лучшем случае только за выбор зарубежного образца, а этот выбор зачастую может быть обусловлен не только техническими характеристиками рассматриваемых образцов. Лучших условий для порождения безответственности трудно даже себе представить, однако они существуют, и никто не собирается изменять существующее положение.

В силу того, что выпускаемые в нашей стране ЭВМ имеют англоязычное математическое обеспечение, разработанное зарубежными фирмами, мы лишены возможности продавать их за границу. Это наносит ущерб и нашей экономике, и нашему престижу.

Возникает резонный вопрос: «А может быть, мы не в состоянии своими силами выйти на мировой уровень вычислительной техники и превысить его?» Ответ здесь однозначный — можем! Подтверждением ему служат разработки отечественных ЭВМ специального назначения.

Ясно одно — необходимы срочные радикальные изменения технической политики в области разработки ЭВМ и их математического обеспечения, возрождения и развития советской школы разработчиков вычислительной техники. Ясно также и то, что вопросы, затронутые в данном письме, выходят за рамки чисто технических проблем.

Уважаемые читатели! На это письмо редакция не подготовила ответа. Просим вас присылать свои соображения по затронутым вопросам. Ответы будут прокомментированы или опубликованы.

Создаются ЭВМ будущего

Завершил разработку элементов компьютеров пятого поколения временный научно-технический коллектив «Старт». Межведомственной комиссией в Академгородке подписан акт о приемке комплекса новейших технических и программных средств.

Первой отечественной системой высокого класса ЭВМ стал действующий лабораторный образец супер-мини-компьютера «Марс-Т». Для решения сложных научно-технических задач предназначен конвейерный вычислитель «Марс-М».

Новое направление в развитии вычислительной техники в нашей стране открывает семейство 32-разрядных микропроцессоров «Кронос». Эти системы признаны оригинальными, существенно опережающими по ряду показателей зарубежные.

— Более двухсот ученых, инженеров и программистов из Новосибирска, Москвы, Таллина, Киева и Северодонецка, — сказал заместитель руководителя «Старта» Е. Кузнецов, — выполнили за три года программу исследований, на которую при традиционных подходах потребовалось бы не меньше пятилетки.

Известия, 14 мая 1988 г.

БАЗОВЫЕ МОДЕЛИ ШКОЛЬНОЙ ИНФОРМАТИКИ

Закончился 3-й учебный год с тех пор, как повсеместно в расписаниях старших классов всех средних школ, а также и СПТУ появился новый предмет «Основы информатики и вычислительной техники». Сегодня уже накоплен некоторый опыт обучения в массовой школе, и потому мы считаем возможным провести коллегиальное обсуждение проблем, которые возникли перед массовой школой за этот период.

В редакции был проведен «круглый стол», участниками которого были авторы, неоднократно выступавшие на страницах сборников, издаваемых издательством «Знание», по вопросам компьютеризации обучения:

— Логвинов Игорь Иосифович — директор Главного информационно-вычислительного центра Министерства просвещения СССР, кандидат педагогических наук;

— Машбиц Ефим Израильевич — заведующий лабораторией Института психологии Министерства просвещения УССР, кандидат психологических наук;

— Ратинский Марк Гершевич — заведующий учебно-методическим кабинетом информатики Московского областного института усовершенствования учителей;

— Шведовский Вячеслав Анатольевич — старший научный сотрудник Института социологических исследований АН СССР, кандидат физико-математических наук.

В этой статье собраны некоторые выступления участников «круглого стола».

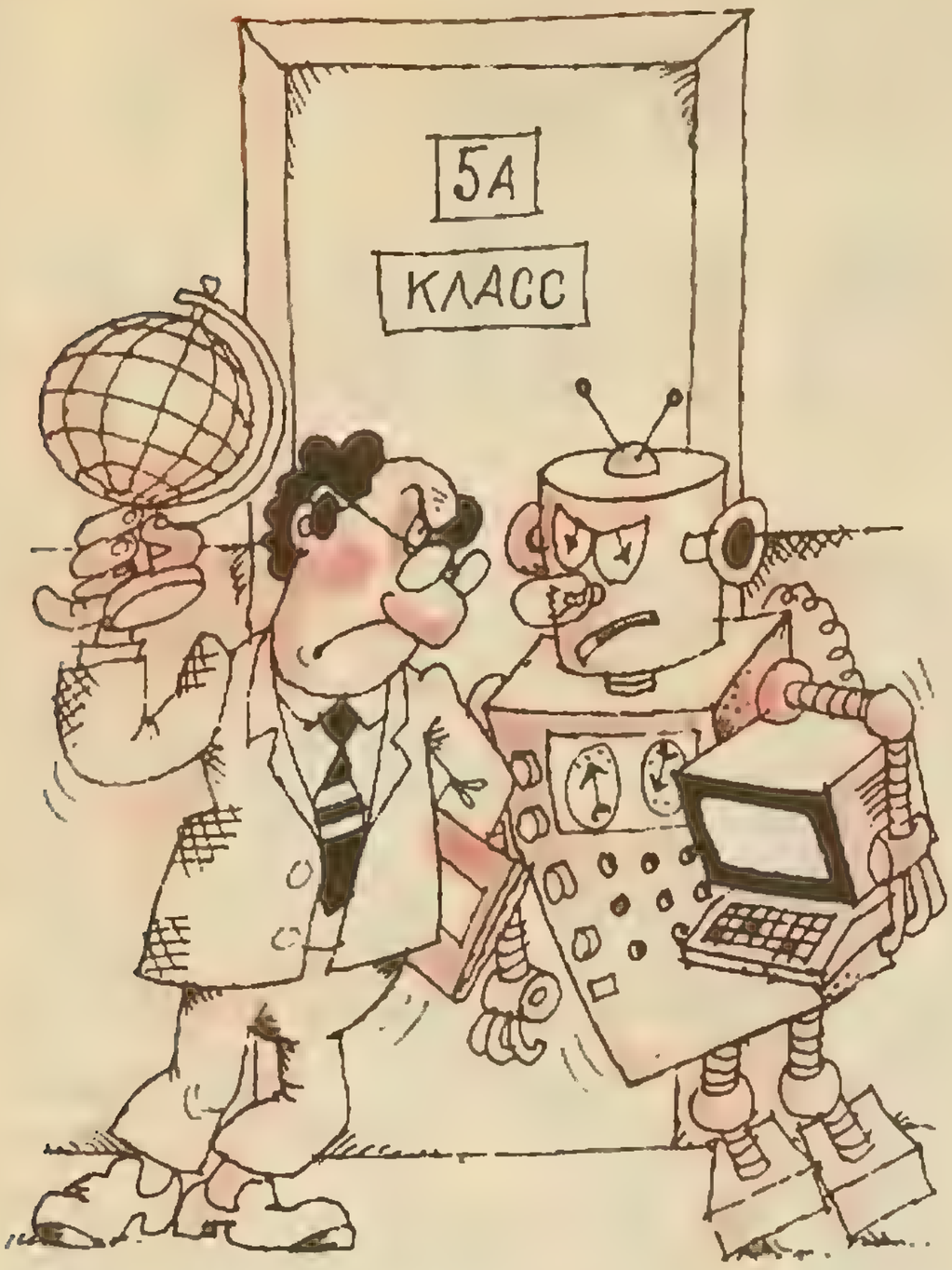
Как подчеркивалось на февральском Пленуме ЦК КПСС, общеобразовательная школа призвана стать подлинно базовым звеном в системе непрерывного образования граждан нашего общества. На пленуме также было подчеркнуто, что в начале тринадцатой пятилетки должна быть решена проблема оснащения всей системы народного образования высококачественной компьютерной техникой.

Именно исходя из этого и учитывая некоторый опыт (как положительный, так и отрицательный), сформулируем первый вопрос.

ЧЕМУ УЧИТЬ?

В. Шведовский. По своему социальному значению информатизация общества вполне сопоставима с его индустриализацией. Поэтому совершенно обоснованно в программу общеобразовательной школы введен новый предмет «Основы информатики и вычислительной техники», который, исходя из стоящих перед школой социальных задач, более корректно именовать «Информатикой». К сожалению, ни пробные учебные пособия, по которым в настоящее время идет обучение в школе, ни рукописи, отмеченные премиями на конкурсе на школьный учебник, который был проведен Минпросом СССР в 1987 г., не отвечают задачам, стоящим перед массовой школой. Однако надо отдать должное мужеству тех, кто сделал первый столь социально важный шаг — создал первый вариант учебной модели информатики для массовой средней школы в нашей стране. И в тех и в других пособиях видно преобладание программирования в ущерб другим разделам информатики. При отработке программы учебного предмета, видимо, следует учитывать такие аспекты, как сложившееся представление о большой информатике в ряде ведущих научных кругов, преподавателей и широкой сети потенциальных пользователей ЭВМ. Игнорирование этого может просто создать отчуждение от курса школьной информатики как учащихся, так и учителей. Следует подчеркнуть то обстоятельство, что массовая школа — это не математическая спецшкола, где подготовка программистов является органической частью обучения, и нам незачем копировать чужие ошибки.

По нашему мнению, в школе при изучении информатики необходимо учить проблемно видеть и ставить задачи, составлять, пусть простейшие, модели явлений и находить математические пути их практического осуществления. При всей своей значимости и программирование, и даже алгоритмирование — это всего лишь часть средств, позволяющих грамотно использовать компьютер. Но цель применения ЭВМ состоит в



решении жизненно важных практических информационных задач.

Е. Машбиц. Процесс компьютеризации обучения необратим, остановить его никто не может. На чем же он основывается?

Во-первых, на возрастающей роли компьютеров в жизни современного общества и, во-вторых, на довольно стремительном росте оснащения компьютерами школ всех регионов планеты.

Следует отметить, что функции компьютера в образовании должны быть разнообразны — от управления самыми различными аспектами учебной деятельности до средств досуга учащихся во внеурочное время. Если же говорить об основных функциях компьютера в учебном процессе, то он выступает и как объект изучения, и как средство обучения. Каждой из этих функций соответствует свое направление компьютеризации обучения. Первое из них предполагает усвоение знаний, умений и навыков, которые позволяют успешно использовать компьютер при решении разнообразных задач, или, другими словами, овладение компьютерной грамотностью. Второе направление видит в компьютере мощное средство обучения, которое способно значительно повысить его эффективность. По нашему мнению, именно эти два направления и составляют основу компьютеризации обучения. Овладение компьютерной грамотностью с психологической точки зрения означает необходимость формирования у человека определенной деятельности. Следует подчеркнуть, что речь идет о новой деятельности, поскольку в ней используются принципиально новые для учащихся средства. И наконец, чтобы обеспечить формирование любой деятельности, а тем более новой, необходимо прежде всего определить, что должно быть получено «на выходе». Другими словами, надо определить цели обучения компьютерной грамоте.

И. Логвинов. Роль и место компьютерной грамотности в системе общеобразовательной подготовки учащихся можно определить, исходя из общей картины внедрения компьютерной техники в разнообразные сферы человеческой деятельности. С учетом глубины и уровня овладения знаниями, умениями и навыками использования компьютерной техники можно условно выделить четыре уровня компьютерной грамотности человека.

Первый — умение достаточно уверенно пользоваться клавиатурой персональной ЭВМ, а также набором готовых средств программного обеспечения для решения задач в своей области деятельности и в быту. В первую очередь это текстовые и графические редакторы, электронные таблицы и т. д.

Второй — овладение (в дополнение к первому уровню) достаточно широкими познаниями в области готовых средств программного обеспечения и методов обработки различных видов информации, умение на этой основе скомпоновать пакет программ, необходимых для решения своей задачи, и спроектировать несложную информационную систему, а также умение использовать консультацию более подготовленного и опытного специалиста и др.

Третий — профессиональная подготовка самых различных направлений: проблемные программисты высокой квалификации, системные программисты, инженеры по обслуживанию и эксплуатации технических средств и др.

Четвертый — это создатели новых ЭВМ, новых программых продуктов широкого использования: операционных систем, систем управления базами данных, новых языков программирования, трансляторов и т. д.

С учетом развития индустрии информатики первым уровнем компьютерной грамотности должны владеть все лица после завершения обучения в общеобразовательной школе и других средних учебных заведениях, т. е. в учебные планы должен быть включен предмет, изучение которого обеспечивает формирование у учащихся знаний, умений и навыков, соответствующих первому уровню компьютерной грамотности.

М. Ратинский. Одним из первичных информационных учреждений является школа, в которой по традиции изучают основы наук, необходимых человеку для дальнейшей трудовой деятельности. Именно с этих позиций, по нашему мнению, и следует рассматривать значение введенного в программу школьного курса в 1985 г. предмета «Основы информатики и вычислительной техники». В силу возрастания роли информатики в жизни современного общества этот предмет призван стать одной из фундаментальных компонентов среднего образования и способствовать устранению некоторых из возникших в школе негативных явлений. Поэтому крайне важно дать, наконец, действительно диалектическое определение самой информатики как науки, а не сводить это определение, как это делается сейчас, к решению ряда, пусть даже и важных, задач на определенном техническом средстве. Если следовать логике авторов этих определений, то они совершенно не допускают, что в недалеком будущем могут появиться информационные системы по принципам их физического устройства, отличные

от современных компьютеров. Для правильного решения возникающих при этом вопросов необходимо провести глубокий научный анализ понятия «компьютерная грамотность», не замыкая это весьма объемное понятие лишь на навыки использования ЭВМ. Необходимо учитывать, что усвоение готовых знаний без овладения деятельностью, в результате которой эти знания получены, формируют догматическое, авторитарное и в конечном случае схоластическое, метафизическое знание. Нам же нужно воспитывать у учащихся диалектическое мышление.

Более того, превращение людей лишь в потребителей знаний означало бы не только потерю ими способностей к продуцированию знаний, но и обесценивание получаемой информации. Все обучение в школе должно быть неразрывно связано с привитием учащимся необходимых элементов культуры.

Информационная культура призвана помочь человеку работать в условиях различных информационных технологий. Проводником этой культуры и призвана стать информатика — наука о структуре и общих свойствах информации. Необходимо отметить, что информатика не начинается, а заканчивается компьютером, который в настоящих условиях представляет собой наиболее современное и действенное средство решения ряда важнейших задач информатики. В школе необходимо обучать цели, а не только одному из средств ее достижения. Исходя из этого компьютер следует прежде всего рассматривать как новое современное средство обучения, которое имеет ряд неоспоримых преимуществ по сравнению с другими традиционными средствами. Введение в процесс обучения компьютеров требует изменения технологии обучения практически по всем предметам и на всех уровнях. Требуется по-новому организовать учебный процесс, что, в свою очередь, ставит очень важный вопрос: какими в этих условиях должны быть учебники и учителя.

Подводя итоги обсуждения первого вопроса, можно сделать вывод о том, что обучение информатике ни в коем случае не должно сводиться лишь к обучению некоторым основам алгоритмизации и программирования. Задача стоит гораздо шире.

Поэтому второй вопрос, требующий раскрытия, — это

«КОГО И КАК УЧИТЬ?».

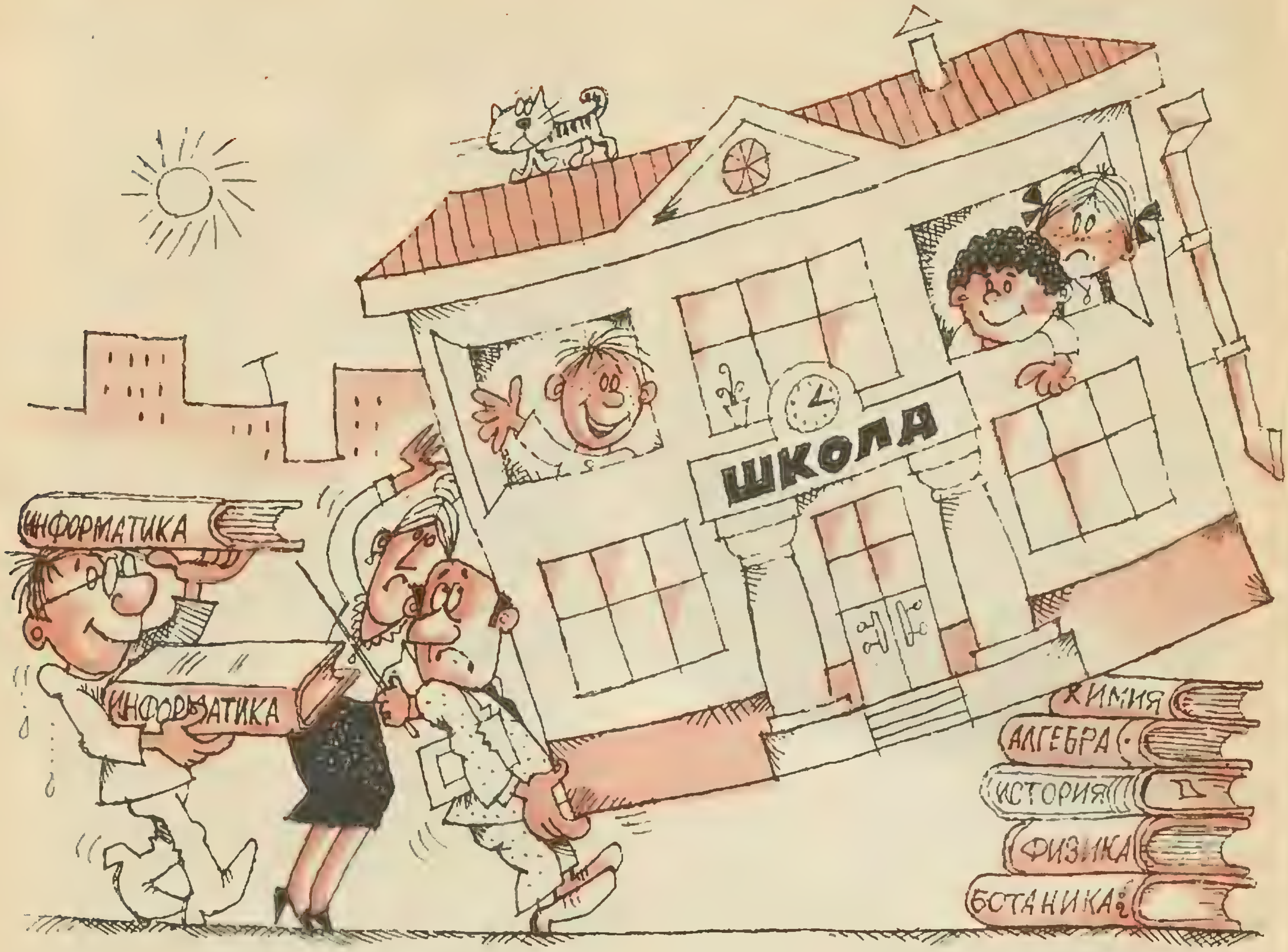
М. Ратинский. Учитель — ключевая фигура любого обучения. Самые совершенные учебные программы и учебники, а также и передовые методики обучения никогда не решат поставленной задачи без должным образом подготовленных учителей. И здесь весьма уместно напомнить слова известного педагога В. А. Сухомлинского: «Воспитывают, обучают не программы, не учебники, не метод, а личность и только личность учителя». Следовательно, основа основ — это целенаправленная подготовка учителей. Речь идет не только об учителях информатики, а о подготовке абсолютно всех учителей к работе в условиях компьютеризации школы. Причем подготовку их следует проводить дифференцированно, поскольку использование компьютера как средства обучения будет иметь весьма существенное различие для учителей различных предметов: истории и математики, физики и литературы, биологии и черчения... В любом случае при подготовке учителей их надо выводить на второй уровень компьютерной грамотности (по классификации И. Логвинова).

Что же касается учащихся, то их следует обучать информатике в более широком аспекте, чем это трактуется рекомендуемыми на сегодняшний день программой и пробными учебными пособиями. По нашему мнению, не совсем верно трактовать компьютерную грамотность лишь с точки зрения общения с компьютером на том или ином уровне. Компьютерную грамотность следует рассматривать как одну из составных частей информационной культуры. Основной задачей школьного курса информатики, по нашему мнению, является развитие у учащихся навыков структурно-логического мышления. Поэтому к весьма существенной проблеме, требующей глубокой проработки и изучения, относится анализ взаимосвязей курса информатики с другими предметами школьной программы, т. е. определение сфер прикладного применения информатики. Эта взаимосвязь не должна ограничиваться лишь составлением программ для решения отдельных задач или циклов задач. По нашему мнению, необходима глубокая перестройка всех курсов школьных предметов на всех уровнях, направленная не только на использование компьютеров в их преподавании, но и на то, чтобы в них готовить учащихся к усвоению самого курса «Информатики».

Е. Машбиц. Исключительно важен с точки зрения психологии вопрос, какую роль будет играть учитель при компьютерном обучении. Останется ли он центральной фигурой учебного процесса, если будет использовать компьютер? Ответить на это можно так. От того, что учитель пользуется учебником, написанным специалистами, учебными пособиями и техническими средствами, изготовленными не им самим, его роль не уменьшается. И в компьютерном обучении от учителя зависит, когда и как использовать обучающую программу, как приспособить ее к определенному контингенту учащихся.

Чтобы эффективно использовать компьютер в учебном процессе, учитель должен приобрести многие знания и умения. И поскольку речь идет о всеобщей компьютерной грамотности, то ею должны овладеть все учителя, а не только преподаватели соответствующих курсов. Все это, разумеется, сопряжено со многими трудностями как психолого-педагогического, так и организационного характера, которые должны быть преодолены.

Для того чтобы педагог мог использовать психологические знания в учебном процессе, необходимо по-новому рассмотреть многие фундаментальные проблемы обучения. В условиях компьютеризации обучения оказалось, что многие психологические и дидактические принципы и концепции просто «не работают», т. е. не могут служить основой для разработки программно-педагогических средств. Следовательно, в основу психологической подготовки педагогических работников следует также положить новейшие данные о закономерностях деятельности обучающего и обучаемого при использовании такого средства обучения, как компьютер.



Останавливаясь на вопросе обучения учащихся, совершенно согласен с тем, что нельзя их обучение сводить лишь к вопросам алгоритмизации и программирования. Школьную информатику следует рассматривать более широко. Именно использование компьютера как средства обучения открывает самые широкие возможности в таком важном аспекте, как индивидуализация обучения, которая весьма ограниченна при традиционной классно-урочной системе, ибо представить каждому учащемуся персонального учителя ни одно общество не в состоянии.

Использование компьютера для индивидуализации обучения в массовом масштабе позволяет осуществить выбор обучающего воздействия (объяснение, подсказка, похвала) с учетом способностей и истории обучения каждого конкретного ученика. Компьютер осуществляет так называемое рефлексивное управление, т. е. строит модель учащегося, которая учитывает особенности его познавательных процессов — воспитания, мышления, памяти и оказывает помощь ученику с учетом его индивидуальных возможностей.

Так же исключительно велики возможности компьютера в проблемном обучении, при котором учащийся выступает как исследователь, самостоятельно открывающий нечто новое. Разумеется, это лишь субъективно новое, уже известное не только в науке, но и в методике. Однако при этом оттачивается ум и воля учащегося, он учится преодолевать трудности, принимать нешаблонные решения.

А. Шведовский. Совершенно согласен, что особую роль в решении столь важного вопроса играет подготовка педагогических кадров. Освоить новую сложную технологию познания, овеществленную в компьютере и его программном обеспечении, или, по выражению философов, распредметить ее для себя может только такая личность, чья внутренняя сложность, динамизм и пластичность, установка на саморазвитие в совокупности адекватно созрели, сформировались, подготовлены к восприятию одухотворенных продуктов творческого труда наших современников — носителей научно-технического прогресса.

Поэтому в подготовке учителей необходимо придерживаться трезвого, реалистического взгляда на то, что компьютеризация обучения — не очередная дань модному веянию, а важнейший социальный заказ общества, призванный обеспечить все расширяющуюся сферу динамики развития научно-технического прогресса.

Обучение учащихся должно быть направлено на привитие таких элементов компьютерной грамотности, которые позволяют подготовить их либо к профессиональному использованию в производстве в качестве рабочего, соприкасающегося с современной микропроцессорной техникой, либо к продолжению образования в учебных заведениях, готовящих специалистов, так или

иначе использующих компьютерную технику в своей деятельности.

И. Логвинов. Мы сможем достигнуть сколь-нибудь значительных результатов в области компьютеризации школьного обучения только в том случае, если она будет рассматриваться как единая программа. И поскольку ключевой фигурой в обучении остается учитель, именно от целенаправленной подготовки учителей к работе в новых условиях и зависит решение этой проблемы. При подготовке учителей надо исходить из следующих обстоятельств. С развитием ЭВМ и расширением областей их применения появились пакеты прикладных программ, приспособленные для решения широких классов задач, системы управления базами данных и пр. Теперь стало возможным использовать для решения задач в самых разнообразных сферах достаточно широкий набор готовых программных средств. Разумеется, это не обеспечивает решение с помощью ЭВМ любой наперед заданной задачи, но существенно изменяет сам характер деятельности на этапе машинного решения задач, который существенно высвобождает человека от рутинного умственного труда.

Обучение учащихся ни в коем случае нельзя сводить к овладению ими некоторыми навыками в программировании и алгоритмизации. Вопрос следует рассматривать гораздо шире. В первую очередь необходимо обучать грамотной постановке задачи. А это означает: уточнение целей и условия их достижения, возможные ограничения на способы и средства достижения целей, уточнение и проверка исходной информации, определение наиболее приемлемых способов представления конечных результатов расчетов. При такой постановке вопроса учащиеся действительно поймут прикладное значение информатики.

Третий вопрос, который представляет интерес:

«С ПОМОЩЬЮ ЧЕГО УЧИТЬ?»

И. Логвинов. Отвечая на этот вопрос, необходимо выяснить: 1. Каким быть учебнику? 2. Какой компьютерной техникой необходимо оснащать школы?

Уже отмечалось, что конкурс на учебник ожидаемого результата не дал. Это было предрешено, поскольку совершенно не учитывались весьма существенные замечания, которые были высказаны по программе, предложенной для написания учебника. Но на этом вопросе мы уже останавливались при ответах на предыдущие вопросы. Второй вопрос заключается в том, какая компьютерная техника должна поставляться в школы. Разумеется, то, что имеется в школах в настоящий момент, не выдерживает никакой критики. А школе необходима вычислительная техника не просто какая-нибудь, а разработанная применительно к школьным условиям. Здесь мы столкнулись с большой проблемой. Проблема школьной ЭВМ обсуждалась во многих инстанциях. Однако вследствие отсутствия педагогического опыта применения вычислительной техники в процессе школьного обучения требования к школьной ЭВМ выдвигали профессионалы-программисты и специалисты вычислительной техники. Подключение к этой работе Минздрава СССР добавило к ним санитарно-гигиенические ограничения. К сожалению, все эти требования в своем большинстве вырабатывались без достаточно обоснованной проработки и обсуждения. Наш опыт по преподаванию информатики в школах позволяет сформулировать некоторые основные требования к школьным компьютерам, выполнение которых в действительности поможет решить задачу широкого использования вычислительной техники в учебном процессе.

1. Компьютеры в обязательном порядке должны быть объединены локальной сетью.

2. Ученический и учительский компьютеры должны быть полностью совместимыми.

3. Память должна быть легко наращиваемой. ОЗУ ученического компьютера в пределах 64—256 К, учительского — 128—512 К (не считая экранной памяти). ПЗУ — встроенное, с одной из базовых систем программирования и текстовым редактором.

4. Клавиатура должна выполняться отдельным блоком и быть особенно надежной. Желательно сменное ПЗУ знакогенератора для национальных школ.

5. Дисплей — в соответствии с требованиями Минздрава СССР. Видеопамять в пределах 32—128 К, что позволит обеспечить отображение информации в режимах: одноцветных символов в строке — 80; цветных — 40—60: графика не менее 256x200x16 (цветов). К учительской ЭВМ необходимо предусмотреть возможность подключения цветного демонстрационного монитора. На рабочем месте ученика допустим монохроматический монитор.

6. Каждое рабочее место ученика должно иметь доступ к внешней памяти учебного комплекта посредством локальной сети.

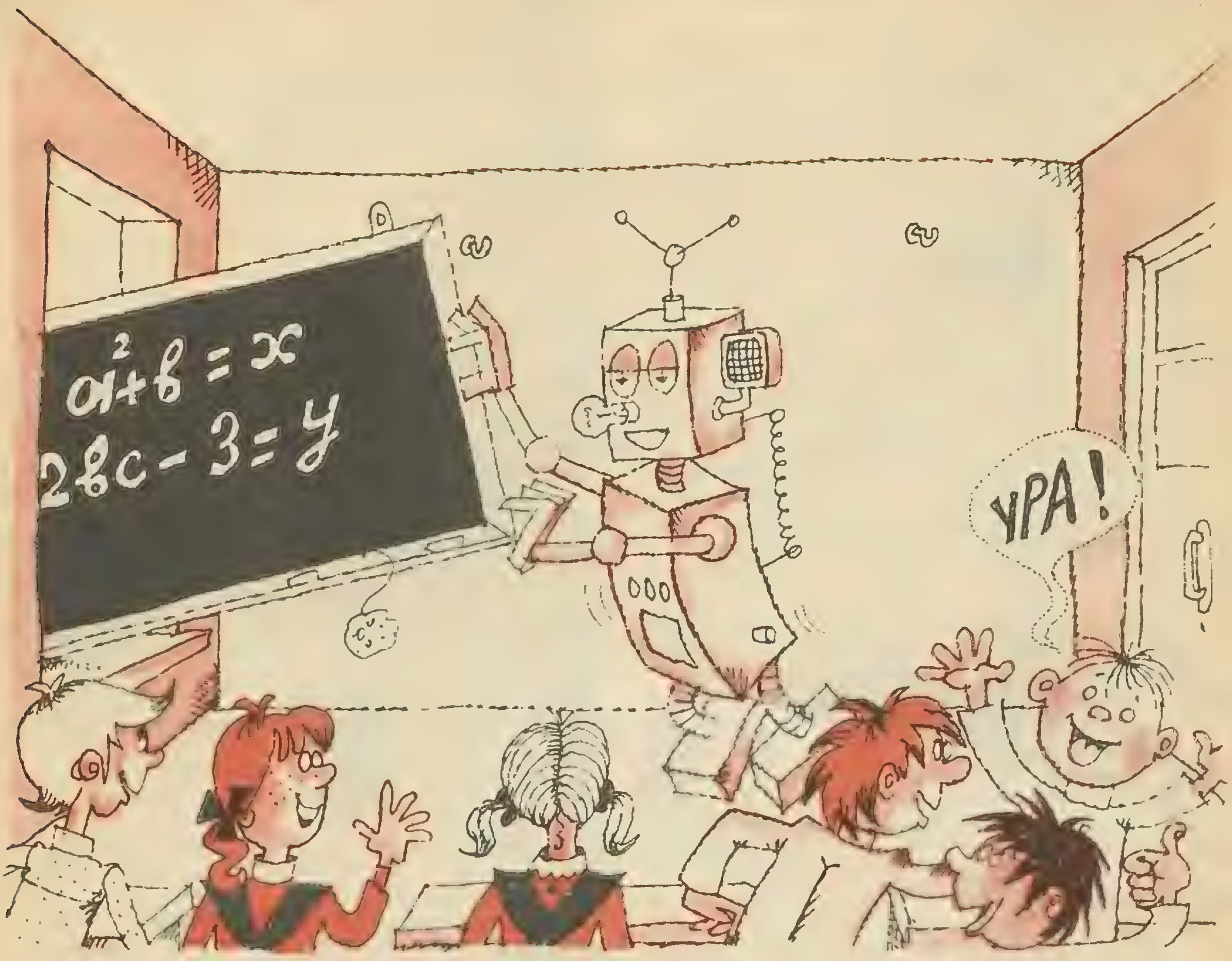
7. Операционные системы должны быть совместимы с широко распространенными. Все программное обеспечение должно работать с 8-битовым кодом символов. Комплект должен быть снабжен пакетами прикладных программ по различным предметам обучения. Для накопления программного обеспечения каждый комплект должен быть снабжен носителями информации емкостью не менее 50 МБ.

Е. Машбиц. Отвечая на поставленный вопрос, особо хотелось бы остановиться на требованиях, которым должны отвечать разрабатываемые программно-педагогические средства.

Основной показатель высокого качества программно-педагогических средств — эффективность обучения. Богатейшие демонстрационные возможности и высокая степень интерактивности системы сами по себе не могут служить основанием для того, чтобы считать обучающую программу полезной. Эффективность программы целиком и полностью определяется тем, насколько она обеспечивает достижение поставленных целей обучения. Все возможности компьютера должны быть проанализированы с точки зрения психологии и дидактики.

Несмотря на то что вопрос эффективности того или иного программно-педагогического средства может быть решен лишь после практической апробации, можно наметить некоторые психолого-педагогические требования, общие для любой обучающей программы.

1. Стимулировать высокую мотивацию уча-



щихся к познанию, причем именно этот аспект не должен развиваться за счет интереса к самому компьютеру.

2. Обеспечить как внешний, так и внутренний диалог.

3. Оказывать содействие при решении учебных задач, обеспечивая педагогически обоснованную помощь с учетом характера затруднения и модели обучаемого, достаточную для того, чтобы не только решить задачу, но и усвоить способ ее решения. Проявлять при этом максимум дружелюбия.

4. Информировать обучаемого о цели обучения, сообщать ему, насколько он продвинулся в ее достижении, его основные недочеты, характер повторяющихся ошибок.

5. Учитывать возможность индивидуализации обучения, позволять учащемуся принимать решение о стратегии обучения, характере помощи и т. д.

6. Адекватно использовать все способы предъявления информации и не навязывать темп ее предъявления.

7. Вести диалог, управляемый не только компьютером, но и обучаемым, позволить последнему задавать вопросы.

8. Позволить обучаемому вход и выход из программы в любой ее точке, обеспечить доступ к ранее пройденному учебному материалу, допускать модификацию, внесение изменений в способы управления учебной деятельностью.

В. Шведовский. Считаю необходимым дополнить изложенное выше следующим. Для школьного обучения необходимо отработать один стабильный вариант компьютерной техники, который по всем своим параметрам не должен уступать аналогичным зарубежным образцам. Необходимо также решить вопрос о выборе языка описания и представления алгоритмов для школьного компьютера. Действующая в настоящий момент языковая разнородность плоха тем, что методические новинки, созданные авторскими коллективами для одного языка, оказываются достоянием только его версий из-за трудностей транслирования на другие языки. Это, безусловно, распыляет усилия ученых, педагогов, практиков. Разумеется, необходимо вооружить обучаемых инструментально, и в этом сильная сторона курса, взявшего на вооружение алгоритмический стиль мышления. Однако инструментальная направленность не должна ограничивать горизонта компьютерного обучения: учащимся надо дать увидеть те великолепные возможности, которые мы получаем и сможем получить при использовании компьютерной техники в различных сферах деятельности человека.

М. Ратинский. Необходимо отметить тот факт, что оснащение школ самой современной компью-

терной техникой без учебника не позволит решить поставленную задачу. Только хороший учебник в сочетании с богатыми возможностями компьютера в действительности позволит реально ощутить значение информатики. Исходя из опыта подготовки учителей для преподавания нового для школы предмета, можно высказать предположение о том, каким должен быть школьный учебник.

В нем должна быть выделена главная мысль о роли информатики в современных условиях, раскрыты основные понятия информатики: информация — алгоритм — компьютер. Материал должен быть изложен языком, доступным для массовой школы.

Разобраны ситуационные задачи с последовательным усложнением по всем основным предметам школьного обучения.

Разъяснено, какую роль играет компьютерная техника в практическом решении ряда задач информатики.

Следует отметить, что, несмотря на то что проведенный конкурс не дал положительных результатов, т. е. не позволил классифицировать ни одну из рукописей в качестве учебника для массовой школы, уже сейчас можно сказать, что рукопись, не представленная на конкурс (авторский коллектив под руководством академика А. П. Ершова), во многом отвечает упомянутым выше требованиям. Замена действующих пробных учебных пособий учебным пособием на основе этой рукописи позволит в значительной мере улучшить обучение в массовых школах по информатике.

И наконец, последний вопрос:

«ДЛЯ ЧЕГО УЧИТЬ?»

М. Ратинский. Компьютеризация школы позволяет совершенно по-новому строить преподавание ряда предметов, поскольку открывает возможности, практически недостижимые для других традиционных средств обучения. Но при всем этом компьютер есть и всегда останется лишь базой данных, и чтобы извлечь из него знания, как из любой другой базы, необходим учитель. И роль учителя в основном должна быть направлена на то, чтобы дать обучаемым необходимые опорные знания, с помощью которых на уровне своих способностей и реальной подготовленности ученик сможет развивать их с использованием компьютера.

Таким образом, сохраняя коллективную подачу основ знаний, мы усиливаем индивидуализацию обучения, что ведет к резкому возрастанию качества знаний обучаемых.

Итак, нет никаких сомнений в том, что введение в школу в качестве одного из основных инструментов для работы с самой разнообразной информацией компьютера — это веление времени, и диктуется оно поступательным движением развития общества.

И. Логвинов. В массовой школе закладываются основные знания у большинства населения страны. Поэтому столь важная в настоящее время социальная задача, как привитие первоначальной компьютерной грамотности, должна решаться именно в школе. Это позволит при правильной постановке обучения информатике не только ознакомить учащихся с некоторыми основами программирования и работой на ЭВМ в качестве оператора, а готовить их к тому, чтобы к окончанию школьного обучения каждый учащийся в достаточной степени знал, где и как с наилучшей отдачей он сможет использовать свой интеллектуальный потенциал, совмещенный с возможностями современных компьютеров. Именно от этого зависит успех внедрения современных технологий на всех уровнях народного хозяйства.

Е. Машбиц. Динамика компьютерного обучения во многом зависит от всех тех, кто прямо или косвенно связан с системой образования.

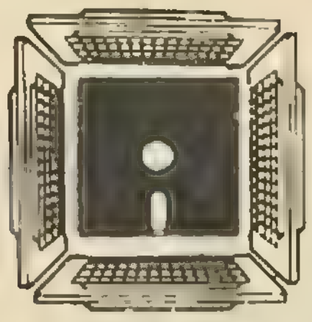
Компьютер — не просто «прибавка» к существующим средствам обучения, не очередная новинка. Он затрагивает все компоненты системы обучения, в том числе и его методы и содержание. В нашей стране имеется хороший теоретический фундамент для построения системы эффективного компьютерного обучения. Это психологические и дидактические теории учебной деятельности разработанные советскими учеными.

Компьютерная грамотность нужна всем — от академика до школьника и от министра до рабочего. Прежде всего она нужна педагогам, причем не только для повышения профессионального мастерства, хотя потенциальные возможности здесь весьма значительны. Всеобщая компьютерная грамотность педагогов — одно из основных условий создания в школе той живительной атмосферы, которая должна обеспечить перестройку школьного обучения в духе педагогики сотрудничества.

В. Шведовский. Информатизация — всеобщий и неизбежный период развития человеческой цивилизации, период освоения информационной картины мира, осознания единства законов функционирования информации в природе и обществе, практического их применения, создания индустрии производства и обработки информации. Современные системы обработки и передачи информации формируют своего рода нервную систему живого организма человеческого общества, придавая этому организму небывалую пластичность и способность к развитию. Человечество, взаимодействуя с окружающей средой, формирует внутри себя в результате процессов отражения и познания информационную модель мира. Разумеется, эффективное участие в столь социально важном процессе невозможно без должных знаний в информатике, основы которых должны закладываться в процессе школьного обучения. И совершенно справедливо академик А. П. Ершов, которого мы по праву считаем основоположником школьной информатики в нашей стране, сформулировал социальную задачу информатизации — от компьютерной грамотности учащихся к информационной культуре общества.

Уважаемые читатели! В выпуске № 4 нашей серии в рубрике «Как убить машинное время» ошибочно помещена программа, не имеющая отношения к тексту описания игры «Кольцо о секретом». Программа, соответствующая этому тексту, будет опубликована в выпуске № 11 серии «Вычислительная техника и ее применение». Редакция приносит извинения за допущенную ошибку.

«ТЕРМИНАЛ»
КОМПЬЮТЕРНЫЙ КЛУБ ШКОЛЬНИКОВ



В прошлом (1987) году в нашей стране были проведены первые Всесоюзные региональные школы для учащихся, увлекающихся решением задач на ЭВМ. Более 500 юных программистов вместе с их гостями, сверстниками из Болгарии, ГДР, Венгрии, ЧССР, Польши, Италии и США, не только овладевали современной техникой, но и пробовали свои силы в создании оригинальных алгоритмов и программ. Опыт первых четырех школ, проводившихся в Сибири (Новосибирск), Прибалтике (литовский город Друскининкай), РСФСР (Переславль-Залесский), в Крыму (малая академия наук «Искатель») и Алма-Ате, оказался удачным, и теперь такие школы будут традиционными, а их число намечено постепенно увеличивать.

Среди творческих результатов участников этих школ нашлось немало настолько интересных и поучительных, что о них следует рассказать и школьникам, и наставникам. Начиная с этого выпуска серии «Вычислительная техника и ее применение», будут публиковаться лучшие работы учащихся. Предпочтение отдано тем из них, в которых решение практически полезных задач основано на неожиданной идее и на оригинальном подходе в использовании ЭВМ.

В качестве основных языков программирования выбраны современные версии БЕЙСИКа и ПАСКАЛЯ. Работы учащихся будут комментировать специалисты.

Формирование этой рубрики поручено Крымской малой академии наук «Искатель». Ответственный — президент МАН «Искатель» доцент В. Н. Касаткин, рецензенты — И. А. Переход и А. С. Металиди.

АЛГОРИТМЫ И ПРОГРАММЫ

«Диофант» — диалоговая система решения и исследования линейных диофантовых уравнений

Андреем Трифоновым (г. Симферополь) разработана диалоговая система «Диофант», с помощью которой можно получить все решения неопределенного линейного диофантова уравнения (ЛДУ): $a_n x_n + a_{n-1} x_{n-1} + \ldots a_1 x_1 = b$,

$a_i, x_i, b \in Z^+$.

Процесс нахождения решений ЛДУ отображается на экране дисплея. В качестве демонстрационной модели, иллюстрирующей алгоритм решения ЛДУ, выбраны русские счеты, измененные так, что в процессе работы число косточек каждого ряда меняется. На таких n-рядных счетах откладываются наборы $x_n, x_{n-1}, \ldots, x_1$, которые можно трактовать как числа, записанные последовательностью цифр x_i, $1 \leqslant i \leqslant n$.

Последовательность всевозможных наборов ограничена снизу набором 0,0, ..., 0, а сверху — набором $l_n, l_{n-1}, \ldots, l_1$, где l_i — верхняя граница (максимальное значение) цифры x_i, причем $l_i = [b/a_i]$.

Некоторые наборы этой последовательности являются решениями ЛДУ. Главная задача — сократить перебор значений каждой цифры x_i, $0 \leqslant x_i \leqslant l_i$ при переходе от набора к набору. В «Диофанте» с этой целью счеты «перекон-струируются» в ходе вычислений. Это означает, что максимальное число

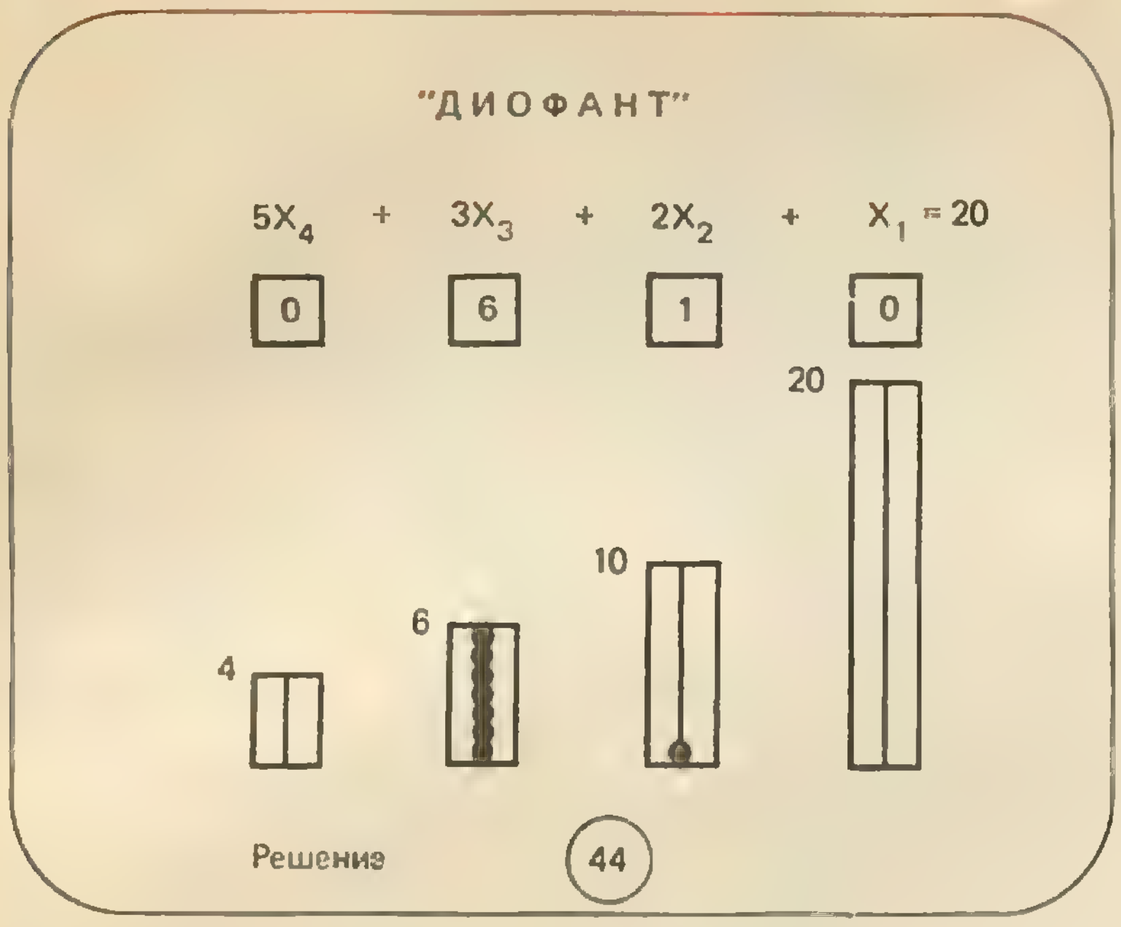


Рис. 1

косточек в i-м ряду не постоянно, оно изменяется в зависимости от величин, установленных в других рядах счет (разрядов числа). Текущее состояние счет, отражающее одно из решений уравнения $5x_4+3x_3+2x_2+x_1=20$, показано на рис. 1.

Начиная с набора 0, 0, ..., b (при $a_1=1$ этот набор есть решение, b — верхняя граница x_1), «Диофант» переходит к очередному набору, «прибавляя» к предыдущему набору единицу. Перенос в «Диофанте» выполняется по правилам, очень похожим на обычное сложение с 1 на счетах или столбиком.

Если единица прибавляется к i-му разряду и i-й разряд содержит максимальное значение (напомним, что оно переменное), то i-разряд устанавливается в нуль, а в разряд $i+1$ переносится 1. Так как значение x_1 всякого предыдущего набора равно верхней границе первого разряда, то при сложении с 1 x_1 обращается в 0, а число $s=a_nx_n+\ldots+a_2x_2$ после сложения с 1 не превышает b. Тогда $[(b-s)/a_1]$ — новая верхняя граница x_1 относительно числа s, представленного всеми разрядами, кроме первого.

Если $[\frac{b-s}{a_1}]=\frac{b-s}{a_1}$,

т. е. если $b-s$ нацело делится на a_1 (для $a_1=1$ это условие всегда выполняется), то, положив $x_1=[\frac{b-s}{a_1}]$,

получим очередное решение ЛДУ.

Например, решение 1,0,0,15, следующее за приведенным на рис. 1, получаем так:

x_4	x_3	x_2	x_1
0	6	1	0
+			1
1	0	0	0

$s=5$, $x_1=20-5=15$.

«Диофант» находит решения ЛДУ в соответствии с основным алгоритмом, приведенным на рис. 2. В программе отсутствует массив верхних границ. При переходе к следующему набору вместо проверки условия $x_i+1>l_i$ проверяется эквивалентное условие $s+a_i>b$, где s — текущее значение набора в процессе суммирования.

Полный перечень (меню) работ, выполняемых системой «Диофант», следующий:

1. Ввод и коррекция исходного ЛДУ.
2. Решение ЛДУ с учетом затраченного на решение времени.
3. Демонстрация процесса вычисления на счетах.

```
10  REM ЛИНЕЙНОЕ ДИОФАНТОВО УРАВНЕНИЕ
20  REM БАЗОВЫЙ АЛГОРИТМ
30  INPUT "N, M";  N, M
40  DIM X (N) , A (N)
50  FOR K=1 TO N: INPUT A (K) : NEXT
60  D= (M–S) /A (1)
70  X (1) =INT (D)
80  IF X (1) <> D THEN 110
90  R=R+1
100 FOR K=N TO 1 STEP–1: PRINT X (K) ; : NEXT :PRINT
110 I=2
120 T=S+A (I) : IF T < =M THEN X (I) =X [I) +1: S=T:GOTO 60
130 S=S–X (I) •A (I)  : X (I) =0 : I=I+1
140 IF I < =N THEN 120
150 PRINT "R="; R
160 END
```

Рис.2

4. Исследование зависимости трудоемкости нахождения решений от *a, b, n.*

В исследовательском режиме «Диофант» реализует численный эксперимент, в ходе которого можно оценить эффективность базового алгоритма. В этом режиме на каждом шаге на экран выводится число разрядов, изменивших свое значение при переходе от предыдущего набора к следующему («расстояние» между соседними наборами). Изменяя некоторые из входных параметров ЛДУ, можно учесть их влияние на трудоемкость решения — в исследовательском режиме «Диофант» обеспечивает вывод графика трудоемкости после решения серии ЛДУ.

«Диофант» реализует диалог через меню и задавая вопросы с заданным множеством ответов. В ходе диалога пользователь задает различные режимы работы с необходимыми уточнениями. В частности, может быть задан тактовый или непрерывный режим вычислений, вывод решений на принтер и т. д.

Post scriptum

Комментарий кандидата физико-математических наук, старшего научного сотрудника И. А. Перехода

Задача решения ЛДУ является одной из основных задач теории дискретной оптимизации. Она имеет большое теоретическое и практическое значение.

К решению ЛДУ и систем ЛДУ сводятся многие прикладные задачи. Примерами таких задач могут служить следующие:

1. На прямой $x+2y=12$ (на плоскости $6x+3y+4z=72$) найти все точки с неотрицательными целыми коэффициентами.

2. Найти все варианты размена двадцатикопеечной монеты монетами достоинством в 1, 2, 3 и 5 копеек.

3. Из данных n предметов различного веса (a_i) и различной цены (b_i) выбрать такие, что их суммарный вес равен k кг, а суммарная стоимость была бы наибольшей (задача линейного программирования).

Актуальна также задача построения хороших алгоритмов решения ЛДУ, т. е. принципиально отличных от полного перебора или максимально уменьшающих его объем.

В основном алгоритме «Диофанта» реализуется метод ветвей и границ, а источником сокращения перебора является динамическое уменьшение границ x_i в ходе вычислений. Качество алгоритма сложным образом зависит от входа задачи, т. е. от параметров *a, b* и *n.* Вопрос о трудоемкости того или иного алгоритма представляет большой интерес, поэтому исследовательские возможности «Диофанта» чрезвычайно полезны.

Пусть ЛДУ имеет k решений. Следовательно, при $a_1=1$ строятся k наборов и каждый очередной набор p', полученный из предыдущего набора p, есть решение ЛДУ. На получение различных наборов затрачивается разное число операций. Расстояние $d=d(p, a, b, n)$ между наборами p и p', т. е. число разрядов, измененных в p, изменяется для различных p.

Очевидно, что $1\leqslant d\leqslant n$. Если перенос завершился в i-м разряде, то $d=i$;

$$x_i:=x_i+1;$$
$$x_{i-1}, \ldots, x_2:=0;$$
$$x_1:=[(a_nx_n+\ldots+a_{i+1}x_{i+1})/a_1].$$

Обратно, если $d=i$, то перенос завершился в разряде i. Поэтому, зная d, можно построить p' по p.

Вычисляя d для каждого p, «Диофант» исследует качество алгоритма, а оно тем выше, чем меньше операций затрачивается на формирование каждого из $N=\sum_{i=1}^{k} d_i$ разрядов, где k — число решений ЛДУ.

В связи с рассмотрением ЛДУ и их систем возникает много задач по математике, информатике и программированию. Назовем некоторые из них.

1. Найти рекурсивную и нерекурсивную формулу (и соответствующие им алгоритмы) для вычисления числа решений ЛДУ без порождения самих решений.

2. Найти функциональную зависи-

```
FUNCTION F (M, N:INTEGER) : REAL;
VAR
   Y : REAL;
   L : INTEGER;
   BEGIN
      Y:=0;
      IF N=1
         THEN
            BEGIN
               Y:=ORD (M MOD A[1]=0);
               IF Y=1
                  THEN
                     BEGIN
                        D[1];=M DIV A[1];
                        WRITESOL
                     END;
            END
      ELSE
   FOR L:=0 TO M DIV A[N] DO
         BEGIN
            D[N]:=L;
            Y:=Y + F (M–A[N]*D[N], N–1);
         END;
   F:=Y
END; (*F*)
```

Рис.3

мость трудоемкости алгоритма решения ЛДУ от входа задачи.

3. Исследовать зависимость d_i от d_{i-1}.

4. Построить рекурсивную функцию решения ЛДУ (см. рис. 3, WRITESOL — процедура вывода очередного решения).

5. Реализовать нерекурсивную процедуру решения ЛДУ, использующую цикл с индексированным параметром вида:

для *x[i]* **от** 0 **до** *b÷a[n]* **нц** ... **кц.**

6. На основе базового алгоритма построить алгоритм решения линейных диофантовых неравенств (ЛДНР), использование которого позволяет решить ЛДУ как частный случай.

7. Построить алгоритмы решения систем ЛДУ и ЛДНР.

8. Построить алгоритм решения задачи линейного целочисленного программирования (поиск экстремума целевой функции на множестве решений ЛДУ и ЛДНР).

9. Найти быстрый алгоритм нахождения $d=d(p, a, b, n)$. (Пусть $A=\{a_n, a_{n-1}, ..., a_1\}$ упорядочено по убыванию, тогда d — «медиана» множества A. Используя упорядоченность A, найти $i_{мин}$, для которого $x_i+1<l_i$. Тогда $d=i$.)

10. Исследовать дерево метода ветвей и границ, используемого в алгоритме решения ЛДУ.

11. Зависит ли $N=\sum_{i=1}^{k} d_i$ от упорядочения A? (Если $N\neq$**const,** то возникает задача нахождения $N_{мин}$.

С учетом перечисленных выше задач система «Диофант» может быть развита в нескольких направлениях. Целесообразно включить в систему, например, обучающий режим, использовать графовые модели для иллюстрации алгоритмов, обеспечить возможность решения систем и задач линейного программирования и т. д.

Перестановка шашек

В работе описывается алгоритм и демонстрационная модель решения головоломки «Перестановка шашек», разработанные Ритой Кармонайте (Литовская ССР).

Суть головоломки: игровым полем является лента, разделенная на $(m+n+1)$ клеток. В левой части игрового поля m клеток заняты белыми шашками, в правой части поля n клеток заняты черными шашками, между белыми и черными шашками одна пустая клетка. На рис. 4 изображено начальное положение.

Необходимо поменять местами белые и черные шашки. Белые шашки можно переставлять только вправо, черные — только влево. Сделать ход — это значит переставить шашку в соседнюю пустую клетку или сделать «скачок» — через шашку иного цвета в пустую секцию.

Введем вспомогательные (рабочие) переменные:

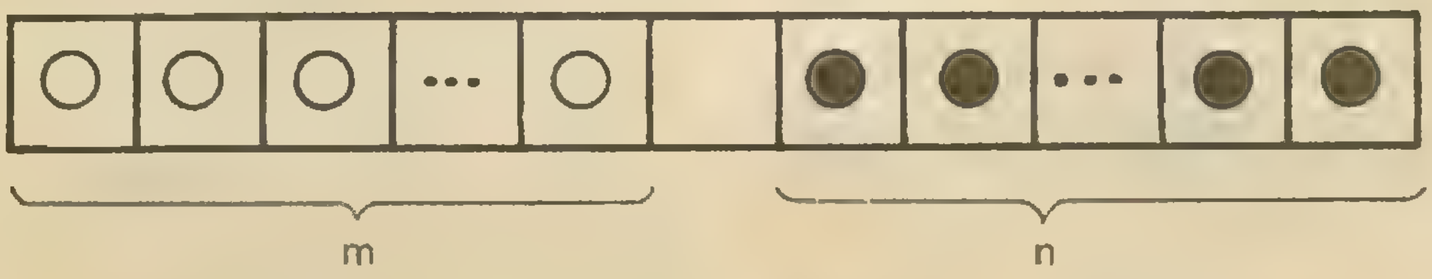


Рис. 4

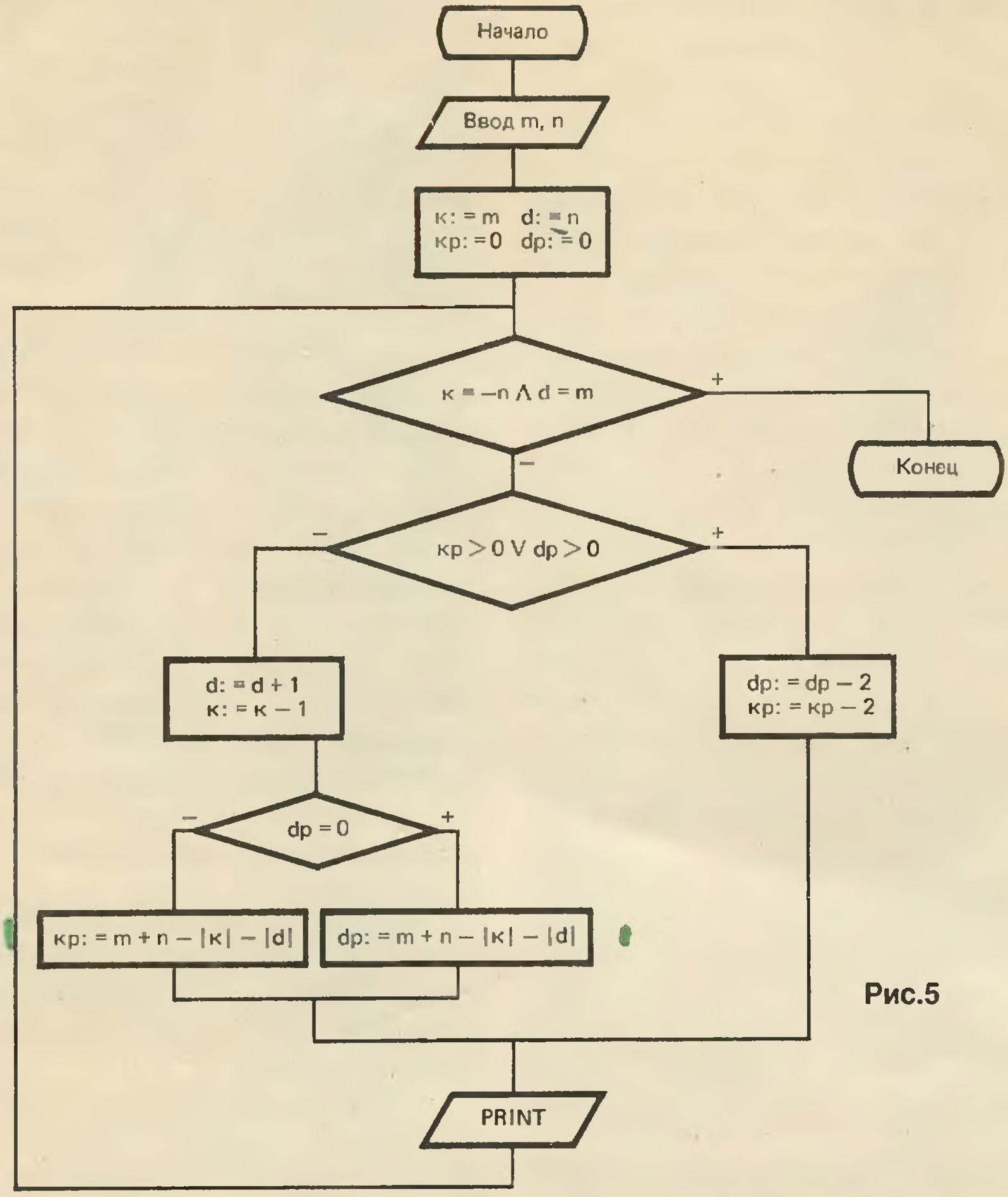


Рис.5

k — число белых шашек, находящихся на левой стороне игрового поля и не имеющих соседей черного цвета. В начале игры *k=m*, в конце *k=–n*;

d — число черных шашек, находящихся на правой стороне игрового поля и не имеющих соседей белого цвета. В начале игры *d=–n*, в конце *d=m*;

kp — число шашек, соединенных в пары и находящихся в левой части поля;

dp — количество пар шашек (в пару входят обязательно шашки разного цвета) в правой части игрового поля.

Вся стратегия игры основана на анализе значений переменных *k, d, kp* и *dp*.

Алгоритм задан схемой, приведенной на рис. 5.

Программа написана на языке программирования ПАСКАЛЬ и не содержит массивов.

Post scriptum

Комментарий инженера-программиста А. С. Металиди

Головоломка, алгоритм решения которой и его демонстрационную машинную модель представила литовская школьница, относится к так называемым динамическим позиционным играм. В этих играх фигуры совершают перемещение на игровых полях.

К таким головоломкам-играм относится головоломка о маневрировании, когда два встречных поезда «расходятся» друг с другом на одноколейном пути, имеющем один небольшой тупик. Задача о переправе козы, волка и капусты с одного берега на другой — еще один пример динамической позиционной головоломки. К этому классу относятся и задачи с шашками, расположенными в один ряд.

```
ROGRAM RITA;
  VAR M, N:INTEGER;
  PROCEDURE P (M, N:INTEGER);
     VAR K, KP, DP, D:INTEGER;
  PROCEDURE PRINT (K, KP, DP, D:INTEGER);
     VAR I:INTEGER;
     BEGIN
        FOR I:= 1 TO ABS (K) DO
           IF K>0 THEN WRITE (' + ') (*WHITE*)
                  ELSE WRITE (' – ') (*BLACK*);
           FOR I:= 1 TO ABS (KP) DIV 2 DO
           IF KP>0 THEN WRITE (' + ')
                   ELSE WRITE (' –+ ');
           WRITE (' ');
           FOR I:=1 TO ABS (DP) DIV 2 DO
           IF DP>0 THEN WRITE (' +– ')
                   ELSE WRITE (' –+ ');
           FOR I:=1 TO ABS (D) DO
           IF D> 0 THEN WRITE (' + ')
                   ELSE WRITE (' – ');
           WRITELN
     END (*PRINT*);
BEGIN
   K:=M;
   KP:=0;
   DP:=0;
   D:=–N;
   PRINT (K, KP, DP, D);
   WHILE NOT ((K=–N) AND (D=M)) DO
      BEGIN
         IF (KP>0) OR (DP>0) THEN (*' +– ' OR ' –+ '*)
            BEGIN
               KP:=KP–2;
               DP:=DP–2;
            END
                              ELSE
            BEGIN
               D:=D+1;
               K:=K–1;
               IF DP=0 THEN KP:=M+N–ABS (K) –ABS (D)
                       ELSE DP:=M+N–ABS (K) –ABS (D)
            END;
         PRINT (K, KP, DP, D);
      END
END;
BEGIN
   READ (M, N);
   WRITELN;
   P (M, N);
END.
```

```
+++   ––––
++  +–––––
++–+   –––
++–+–    ––
++–   –+––
+  –+–+–––
  +–+–+–––
––+  +–+––
––+–+  +––
––+–+–+  –
––+–+–+
––+–+   –+
––+–   –+–+
––   –+–+–+
––––  +++
```

```
+++   ––––
++  +–––––
++–+   –––
++–+–    ––
++–   –+––
+  –+–+–––
  +–+–+–––
––+  +–+––
––+–+  +––
––+–+–+  –
––+–+–+
––+–+   –+
––+–   –+–+
––   –+–+–+
––––  +++
```

В рассмотренной выше головоломке в каждой позиции допустимо не более двух ходов, известно, что для таких игр возможен полный анализ задачи. Достаточно, например, построить двоичное дерево игры и проанализировать все его «ветви». Можно подсчитать, что оптимальная стратегия решения головоломки требует всего $(mn+m+n)$ ходов. При этом $(m \cdot n)$ ходов являются «скачками», а $(m+n)$ ходов «сдвигами».

Школьница проявила изобретательность в построении формализованной модели: удачно найдены рабочие переменные *k*, *o*, *kp* и *op*, с помощью которых ясно описывается ситуация, сложившаяся на «игровом поле», и дается рекомендация о том, какой делать ход в любой ситуации. Найденный алгоритм оптимален.

Текст программы на языке ПАСКАЛЬ выписан в соответствии с современными требованиями. Нам представляется, что в текст следует внести одно изменение. Речь идет о распечатке «протокола» игры так, чтобы в каждой строке вместо символов „+“ и „–“ были изображены «кружочки» белые и черные.

У 59 **Универсальный** язык — возможно ли это? Сб. статей. М.: Знание, 1988. — 48 с. — (Новое в жизни, науке, технике. Сер. „Вычислительная техника и ее применение“; № 8).

15 к.

В брошюре, основываясь на высказываниях многих ученых различной специализации, обсуждена проблема создания универсального языка программирования. Здесь же другой коллектив ученых (педагогов и психологов) обсуждает вопросы преподавания в школе курса информатики и вычислительной техники.

Рассчитана на широкий круг читателей.

2404010000 **ББК 32.973—01**

Т·Е·М·А СЛЕДУЮЩЕГО ВЫПУСКА:

ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Научно-популярное издание

УНИВЕРСАЛЬНЫЙ ЯЗЫК — ВОЗМОЖНО ЛИ ЭТО?

Гл. отраслевой редактор *Л. А. Ерлыкин*
Редактор *Б. М. Васильев*
Мл. редактор *Н. А. Васильева*
Художники *В. Н. Конюхов и К. Н. Мошкин*
Худож. редактор *М. А. Гусева*
Техн. редактор *Т. В. Луговская*
Корректор *В. И. Гуляева*

ИБ № 9230

Сдано в набор 22.04.88. Подписано к печати 27.06.88. Т 08376. Формат бумаги 70×100$^{1}/_{16}$. Бумага офсетная № 2. Гарнитура гельветика. Печать офсетная. Усл. печ. л. 3,90. Усл. кр.-отт. 8,45. Уч.-изд. л. 4,56. Тираж 65820 экз. Заказ 2079. Цена 15 коп. Издательство «Знание». 101835, ГСП, Москва, Центр, проезд Серова, д. 4. Индекс заказа 884708. Ордена Трудового Красного Знамени Калининский полиграфический комбинат Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли. 170024, г. Калинин, пр. Ленина, 5

Цена 15 коп.

Индекс 70195

Адрес подписчика:

Издательство «Знание»

Подписная научно-популярная серия

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Дорогой читатель!

Брошюры этой серии в розничную продажу не поступают, поэтому своевременно оформляйте подписку.
Подписка на брошюры издательства «Знание» ежеквартальная принимается в любом отделении «Союзпечати».

Наш адрес:
Москва,
Центр,
проезд
Серова,
д. 4